MASSACHUSETTS INSTITUTE OF TECHNOLOGY VLSI PUBLICATIONS

VLSI Memo No. 89-545
May 1989

Nearly Optimal Algorithms and Bounds for Multilayer Channel Routing

Bonnie Berger, Martin Brady, Donna Brown, and Tom Leighton

Abstract

This paper presents algorithms for routing channels with $L \geq 2$ layers. For the unit vertical overlap model, we describe a two-layer channel routing algorithm which uses at most $d + O(\sqrt{d})$ tracks to route two terminal net problems and $2d + O(\sqrt{d})$ tracks to route multiterminal nets. We also show that $d + \Omega(\log d)$ tracks are required to route two-terminal net problems in the worst case even if arbitrary vertical overlap is allowed. We generalize the algorithm to unrestricted multilayer routing and use only $d/L\text{-}1 + O(\sqrt{d/L} + 1)$ tracks for two-terminal nets (within $O(\sqrt{d/L} + 1)$ tracks of optimal) and $d/L\text{-}2 + O(\sqrt{d/L} + 1)$ tracks for multiterminal net problems (within a factor of $L\text{-}1/L\text{-}2$ times optimal). We also show that our general routing strategy can be used to duplicate the best upper bounds for the Manhattan and knock-knee routing models, and gives a new upper bound for routing with 45-degree diagonal wires.

Microsystems Research Center Room 39-321
Massachusetts Institute of Technology
Cambridge Massachusetts 02139
Telephone (617) 253-8138

Acknowledgements

This research was supported in part by the National Science Foundation under contract number ECS-8352185, the Air Force under contract number AFOSR-86-0078, Defense Advanced Research Projects Agency under contract numbers N00014-80-C-0622 and N00014-87-K-0825, an NSF Presidential Young Investigator Award with matching funds from Xerox Corporation, International Business Machines Corporation, and AT&T Bell Laboratories.

Author Information

Berger: Laboratory for Computer Science, Room NE43-328, MIT, Cambridge, MA 02139. (617) 253-5889.

Brady: Lockheed R&D Division, O-9740/B-202, 3251 Hanover Street, Palo Alto, CA 94304.

Brown: Coordinated Science Laboratory, University of Illinois, Urbana, IL 61801.

Leighton: Department of Mathematics and Laboratory for Computer Science, Room NE43-332, MIT, Cambridge, MA 02139. (617) 253-5876.

Nearly Optimal Algorithms and Bounds for Multilayer Channel Routing

Bonnie Berger[1], Martin Brady[2], Donna Brown[3], Tom Leighton[4]

[1] Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139

[2] Lockheed R&D Division, O-9740/B-202
3251 Hanover St.
Palo Alto, CA 94304

[3] Coordinated Science Laboratory
University of Illinois
Urbana, IL 61801

[4] Mathematics Department and
Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139

Abstract. This paper presents algorithms for routing channels with $L \geq 2$ layers. For the unit vertical overlap model, we describe a two-layer channel routing algorithm which uses at most $d + O(\sqrt{d})$ tracks to route two-terminal net problems and $2d + O(\sqrt{d})$ tracks to route multiterminal nets. We also show that $d + \Omega(\log d)$ tracks are required to route two-terminal net problems in the worst case even if arbitrary vertical overlap is allowed. We generalize the algorithm to unrestricted multilayer routing and use only $\frac{d}{L-1} + O(\sqrt{d/L} + 1)$ tracks for two-terminal nets (within $O(\sqrt{d/L} + 1)$ tracks of optimal) and $\frac{d}{L-2} + O(\sqrt{d/L} + 1)$ tracks for multiterminal net problems (within a factor of $\frac{L-1}{L-2}$ times optimal). We also show that our general routing strategy can be used to duplicate the best upper bounds for the Manhattan and knock-knee routing models, and gives a new upper bound for routing with 45-degree diagonal wires.

This research was supported in part by NSF ECS-8352185, Air Force Contract AFOSR-86-0078, DARPA contract N00014-80-C-0622, an NSF Graduate Fellowship, an NSF Presidential Young Investigator Award with matching funds from Xerox, IBM, and AT&T.

1. Introduction

Channel routing plays an important role in the development of automated layout systems for integrated circuits (e.g. see [HS71], [Ri82]). Many layout systems first place modules on a chip and then wire together terminals on different modules that should be electrically connected. This wiring problem is often solved by heuristically partitioning the given space into rectangular channels and then assigning to each such channel a set of wires which are to pass through it. This solution reduces a "global" wiring problem to a set of disjoint (and hopefully easier) "local" channel routing subproblems. For this reason, the channel routing problem has been intensively studied for over a decade, and numerous heuristics and approximation algorithms have been proposed for its solution ([BBL84], [BB86], [De76], [PL84], [RBM81], [RF82], [YK82]).

The generic form of the channel routing problem may be described as follows. The *channel* consists of a rectilinear grid of *tracks* (or rows) and *columns*. Along the top and bottom tracks are numbers called *terminals*, and terminals with the same number form a *net*. A net with q terminals is called an *q-terminal net*. The smallest net is a *two-terminal* net; if $q > 2$, we have a *multiterminal* net. The channel routing problem is to connect all the terminals in each net using horizontal and vertical wires which are routed along the underlying rectilinear grid. The goal is to complete the wiring using the minimum number of tracks; i.e., to minimize the *width* of the channel.

A variety of models have been proposed for channel routing, with differences depending on the number of layers allowed and on the ways in which wires are allowed to interact. The simplest is *river routing*, in which only one layer of interconnect is available. Unfortunately, only planar problems are routable, and even a simple routable problem (like an N-net shift-left-by-1) may require N tracks to river route.

Two-layer models are more relevant in practice, and have been extensively studied. One of the most common two-layer models is the traditional *Manhattan* routing model. In the Manhattan model, all horizontal wire segments are routed in one layer and all vertical wire segments are routed in the other layer. Hence, wires can *cross* but cannot *overlap* (i.e., run on top of one another) for any distance. Note that when a wire changes direction, it must also change layers, which requires a *contact cut* at the corresponding grid point. Where a contact cut is used, no other electrically disjoint wire can pass through that grid point in either layer.

Unfortunately, channel routing is NP-complete for many interesting routing models, including Manhattan routing, even when restricted to two-terminal nets ([Sz85], [SY82]). However, a linear time approximation algorithm is known for Manhattan routing [BBL84]. The bounds obtained for channel width are based on the notions of *density*, d, and *flux*, f. The density of a channel routing problem is the maximum number of distinct nets crossing (or touching) any vertical cut of the channel. It is not difficult to see that the density of a problem is a lower bound on its channel width in the Manhattan model. A channel has flux f if f is the largest integer for which some horizontal cut spanning $2f^2$ columns splits at least $2f^2 - f$ nets. (The flux definition assumes no *trivial* nets, i.e., no two-terminal nets with both terminals in the same column). Flux, like density, is a lower bound on channel width [BBL84] and although the flux can be as large as $\sqrt{N}/2$ for an N-net problem (e.g., the shift-left-by-1), it is often much smaller in practice. In [BBL84], Baker, Bhatt and Leighton devised an algorithm to route any Manhattan problem with density d and flux f in a channel of width $2d + O(f)$. For two-terminal nets, the upper bound is $d + O(f)$. Recently, the multiterminal net bound was improved to $3d/2 + O(\sqrt{d \log d}) + O(f)$ by Gao and Kaufmann [KG87].

The *knock-knee* model proposed by Rivest, Baratz, and Miller [RBM81] also uses two layers, but does not constrain vertical wires to be routed in a different layer than horizontal wires. Hence, wires are allowed to share corners (i.e., a wire can bend on the top layer directly above a wire bending on the lower layer), but they are still not allowed to overlap. Density again serves as a lower bound on channel width in the knock-knee model. Flux does not play a role, however, and every two-terminal net knock-knee problem can be routed using $2d-1$ tracks [RBM81], [BB82] ($3d + O(\sqrt{d \log d})$ tracks are sufficient for multiterminal net problems, by the new results of Gao and Kaufmann [KG87]). Somewhat surprisingly, $2d-1$ is actually optimal in the worst case; this lower bound was demonstrated by Leighton [Le82] when he discovered a class of two-terminal net problems which require $2d-1$ tracks for any d.

The question remained, however, whether a better algorithm could be found if a less restrictive (but still electrically sound) model were allowed. In the first part of this paper, we answer this question affirmatively by describing an algorithm that routes any two-terminal net problem with density d using only $d + O(\sqrt{d})$ tracks. For multiterminal net problems, the upper bound is $2d + O(\sqrt{d})$. The model used is an extension of the knock-knee model, that we call the *unit vertical overlap* model, in which wires are allowed to overlap for unit segments in the vertical direction. In independent work, Gao and Hambrusch established $d + O(d^{2/3})$ and $2d + O(d^{2/3})$ bounds for routing in the unit vertical overlap model [GH86]. By extending the algorithm in this paper, Gao and Kaufman [GK87] have subsequently improved the multiterminal net bound to $3d/2 + O(\sqrt{d \log d})$ tracks.

Density is, of course, an obvious lower bound on channel width in the unit vertical overlap model. In addition, we present an improved lower bound for two-layer routing. We show that some two-terminal net CRPs require $d + \Omega(\log d)$ tracks even when vertical overlap of arbitrary length is allowed. This is significant in that it shows that optimal routing algorithms must use $d + \Theta(g)$ tracks in the worst case, where g is some increasing function of d.

While the two-layer algorithm is intended primarily to lay the groundwork for the multilayer algorithm, the result is significant in its own right for several reasons. First, though the model was weakened, a factor of two improvement was obtained, which is very important in practice. Second, it is the last factor of two that can be gained; we are approaching the lower bound. Third, the result shows that some routing problems become easier when unit vertical overlap is allowed. Since short lengths of overlap may be allowable in practice, the result may lead to innovations in practical routing algorithms. Fourth, and possibly most important, our algorithm presents a unified framework within which we can rederive previous results for Manhattan and knock-knee routing, and obtain new results for routing with 45-degree diagonal wires. This is somewhat surprising since the previous algorithms were very different and highly model-dependent. It is important to identify algorithms that are tolerant to variations in model restrictions, since real-world channel routing problems often differ from the corresponding mathematical abstractions. Hence, algorithms that are tolerant to model changes are more likely to be reducible to practice in the long run.

In the second part of this paper, we describe the extension of our algorithms to multilayer channel routing. Recent advances in fabrication technology have increased the importance of multilayer channel routing. The initial theoretical work in this area is due to Preparata and Lipski [PL84] who defined the three-layer knock-knee model and discovered an efficient algorithm for routing any two-terminal net problem in the optimal number of tracks, d. For routing multiterminal nets in this knock-knee three-layer model, the best known algorithm uses $3d/2 + O(\sqrt{d \log d})$ tracks [GK87].

Some recent results have been obtained for routing with $L \geq 3$ layers. Hambrusch in [Ha85] gave an algorithm which routes two-terminal nets using $\frac{d}{\lceil \frac{L-1}{2} \rceil} + 3$ tracks for $L \geq 5$ with arbitrary overlap; $\frac{2}{3}d$ and $\frac{3}{4}d$ results are obtained for L=4 and L=3, respectively. Subsequently, Brady and Brown [BB86] used arbitrary overlap to route any multiterminal net problem using at most $\frac{d+2}{\lfloor \frac{2}{3}L \rfloor}$ +5 tracks for $L \geq 7$ (inferior results are obtained for smaller L values). For a different model, in which wires are allowed to overlap, but not on adjacent layers, their algorithm uses at most $\frac{2d}{L}$ +3 tracks, within three tracks of optimal.

Extending Leighton's lower bound of d for L=2 [Le82], Hambrusch showed that at least $\left\lceil \frac{d}{L-1} \right\rceil$ tracks are required to route some channel routing problems with L layers, even if wires are allowed to overlap [Ha83]. In this paper we shall generalize our two-layer algorithm in order to route any two-terminal net problem using $\frac{d}{L-1} + O(\sqrt{d/L} + 1)$ tracks and using $\frac{d}{L-2} + O(\sqrt{d/L} + 1)$ to route any multiterminal net problem. Notice that these results are very close to optimal in the worst case: within $O(\sqrt{d/L} + 1)$ tracks for two-terminal nets and within a multiplicative factor of $\frac{L-1}{L-2}$ times optimal for multiterminal nets.

The remainder of this paper is divided into sections as follows. In Section 2, we present the two-layer unit vertical overlap algorithm. In Section 3 we briefly explain how to replicate the known results for Manhattan and knock-knee routing, and obtain a better upper bound for routing with 45-degree diagonal wires using the same strategy. We generalize the algorithm to multiple layers in Section 4. In Section 5 we improve upon the lower bound for overlap routing. We conclude in Section 6 with some remarks and open questions.

2. The Two-Layer Unit Vertical Overlap Model

2.1. Two-Terminal Nets

First we shall describe the routing algorithm in detail for the two-layer unit vertical overlap model, allowing only two-terminal nets. Routing in the various other models is subsequently described by extending this basic algorithm. The algorithm proceeds in two phases. In the first phase, we start by partitioning the channel into *blocks* of r consecutive columns. Tracks in the *middle section* of the channel are then used to route each net from the block in which the net begins to the block in which it ends. The second phase completes the routing, using the *top* and *bottom sections* of the channel to route the nets into the correct columns within each block (see Figure 1).

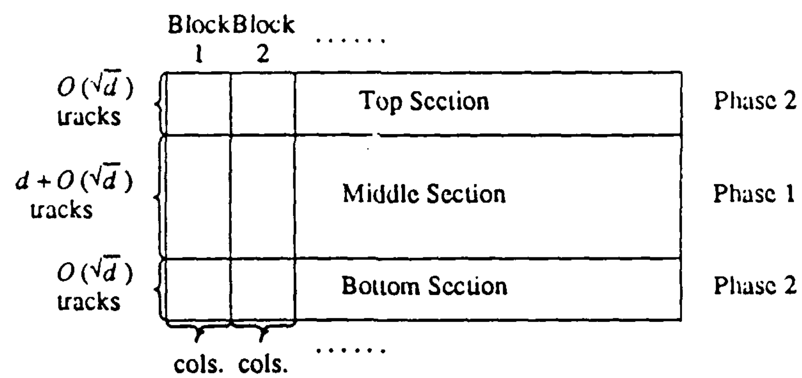


Figure 1. The two phase partition of the unit vertical overlap algorithm.

Phase 1 is the core of the routing strategy. We distinguish three different types of nets in Phase 1. A *vertical* net has both of its terminals in the same block. Among nets having terminals in different blocks, a *rising* (*falling*) net has its rightmost terminal on the top (bottom) of the channel. We further classify the rising and falling nets as follows. A *starting* (*ending*) net in a block B has its leftmost (rightmost) terminal in B. A net having terminals in blocks both to the left and to the right of block B is called a *continuing* net in B. Phase 1 proceeds a block at a time from left to right, employing a *rising/falling* strategy: within the middle section, rising (falling) nets are packed into the highest (lowest) tracks available for horizontal routing. The empty tracks between the rising and falling nets are reserved to route starting nets.

Phase 1 is further subdivided into two parts, Phases 1.1 and 1.2. Phase 1.1 is very similar to the Manhattan interblock routing procedure presented in [BBL84]. Ending nets are routed in the leftmost columns of the block, followed by the starting nets, and finally vertical nets. Falling (rising) ending nets are exited in order from the highest track to the lowest (lowest to the highest). The highest falling (lowest rising) continuing nets are then routed into the vacancies in order to maintain the rising/falling structure. Finally, entering falling (rising) nets are routed into the lowest (highest) empty routing tracks. This forms a natural staircase pattern, as can be seen by looking ahead to Figure 3.

In order to avoid collisions between rising and falling nets entering in the same column, nets may need to *backtrack* to the left before being routed into their target track. A *pyramid* structure of empty routing paths, defined by [BBL84], is maintained between the falling and rising tracks for this purpose (see Figure 2a). The pyramid contains all of the empty routing tracks which are currently available for routing new starting nets. As ending nets vacate tracks, the pyramid is expanded in order to reclaim these tracks, as in Figure 2b. Starting nets are routed through the outermost tracks, directly from one side and backtracking through the other, so that a contracted pyramid remains in the center of the channel (see Figure 2c). If the number of empty tracks on either side of the pyramid becomes unbalanced (as was the case in Figure 2c) the pyramid must be *reshaped*. Examples of reshaping are given in Figures 2d and 2e.

In [BBL84], three empty columns are required in each block in order to separate the staircases of ending, continuing, and starting nets routed in this phase, leading to an additive flux term. Here, Phase 1.1 assumes that five empty tracks, called *good tracks*, G1 - G5, have been placed in the correct positions upon entering the block. These will serve the same purpose as the empty columns in the routing procedure of [BBL84]. Two good tracks are positioned on either side of the pyramid, serving as buffer tracks between the pyramid and the rising and falling nets. A fifth good track is positioned in the center of the pyramid to separate the starting rising and falling nets. Each good track used results in an empty track at the right of the block, called a *bad* track. The purpose of Phase 1.2 is to move the five empty bad tracks from the previous block into the correct good track positions for use in the next block.

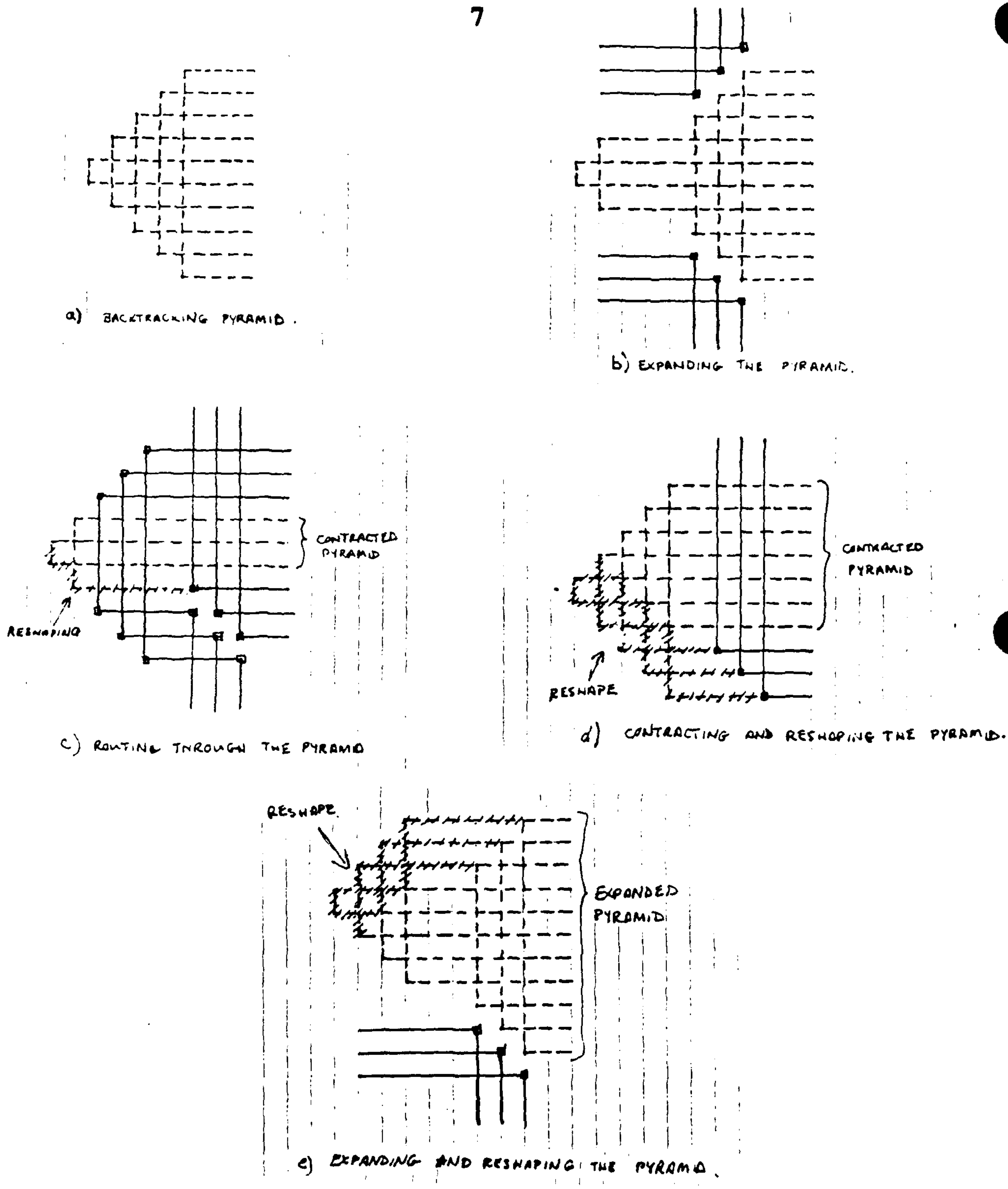


Figure 2. Maintaining the backtracking pyramid.

It is important to note that Phases 1.1 and 1.2 do not separately route different portions of the channel. Rather, Phase 1.1 determines some of the data for the routing of a block, Phase 1.2 then determines the rest. The information from these two phases is then combined to determine the physical placement of wires in the middle section of the current block. These two phases can be considered independently, and when we speak, for instance, of Phase 1.1 *routing*, we mean the set of decisions made in Phase 1.1.

We now outline Phase 1.1 in detail (refer to Figure 3 for examples). The Phase 1.1 portion of the routing is done Manhattan style; vertical wires run in layer 1, horizontal wires in layer 2. Where possible, we describe only the falling net portion of the routing; the rising nets are handled symmetrically. For simplicity, we assume that all nets are *top-to-bottom nets*, i.e. that all nets have one terminal at the top of the channel and one terminal at the bottom of the channel. The extension of the algorithm to handle *same-side nets* (both terminals are on the same side of the channel) is straightforward, since such nets never have to cross the channel.

Step 1: Route the ending nets.

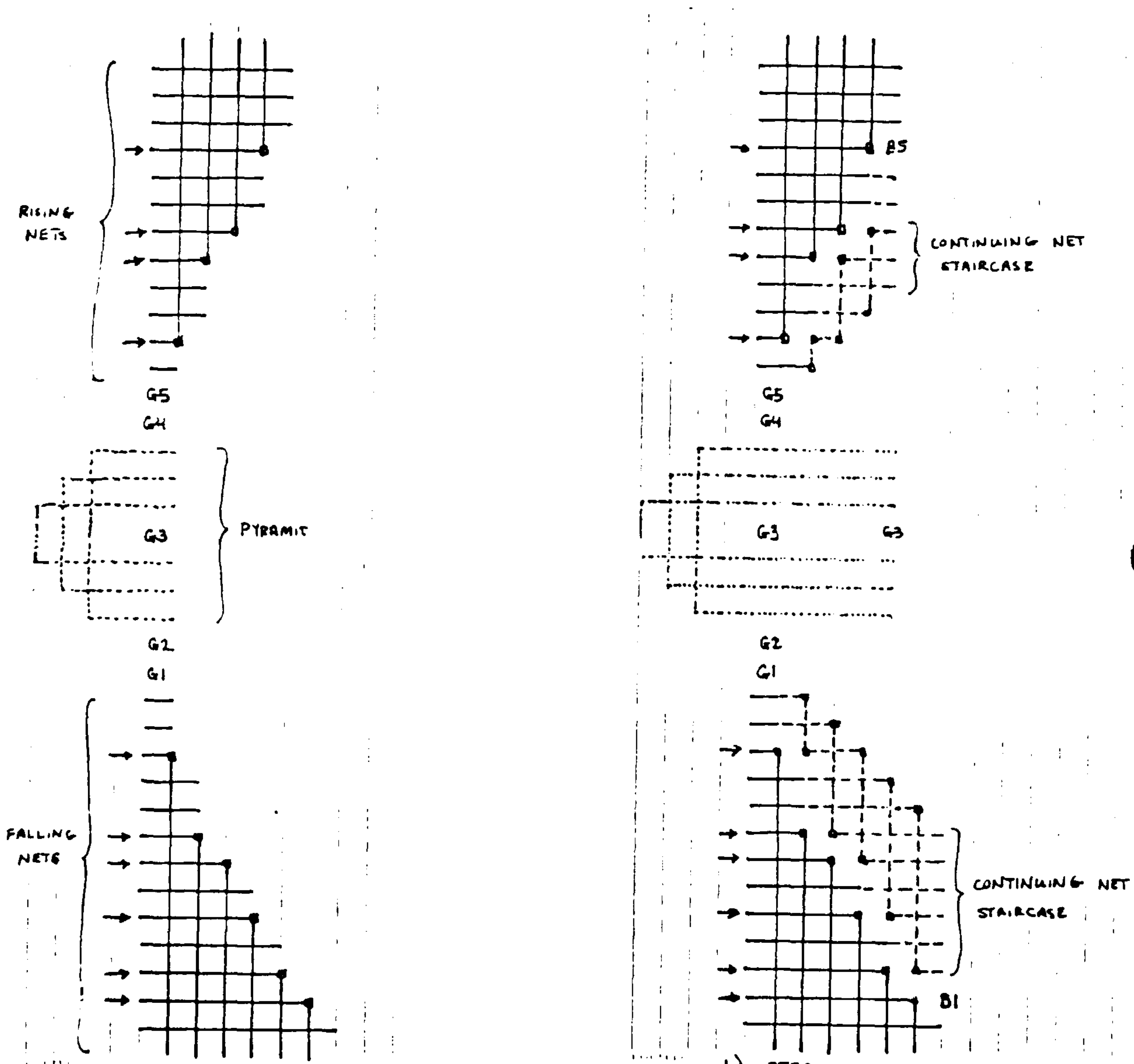
All falling nets ending in B exit in the leftmost columns, ordered from the highest to lowest. A similar strategy is employed on the rising nets ending in B in the upper part of the channel. In this case, rising nets are exited from lowest to highest. This produces a *staircase pattern* of ending nets, leaving empty vacated tracks scattered through the falling portion of the channel (see Figure 3a).

Step 2: Fill in vacated tracks with continuing nets.

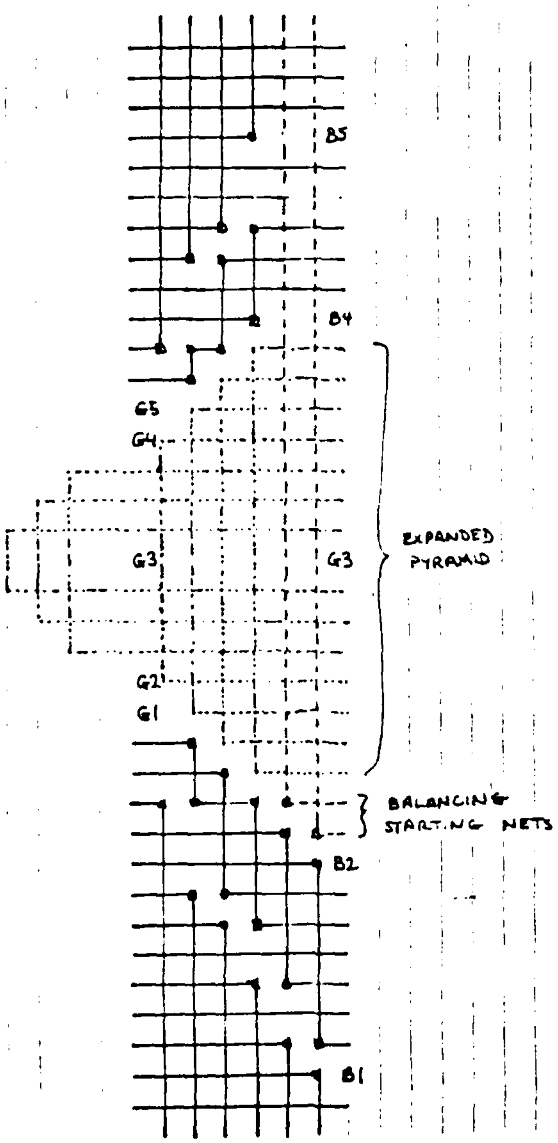
Tracks vacated in Step 1 are filled in using falling continuing nets, so that the falling nets remain packed into the bottom tracks. Falling continuing nets are routed in order from the top down into the vacated tracks (see Figure 3b). Notice that the track vacated in the first column cannot be filled in until the second column. However, the continuing nets must begin vacating the topmost tracks immediately, so that the pyramid can be expanded into the vacated tracks. G1 solves this problem by serving as the highest falling track, already vacated. Each subsequent falling continuing net can then be routed on schedule, forming the continuing net staircase. The lowest track vacated in Step 1 is not filled in, and becomes bad track B1, corresponding to good track G1. The falling and continuing net staircases are thus separated by using G1→B1 as a buffer track. A symmetric algorithm is employed at the upper end of the channel, using G5→B5 as a buffer track.

Step 3: Balance the falling and rising net columns.

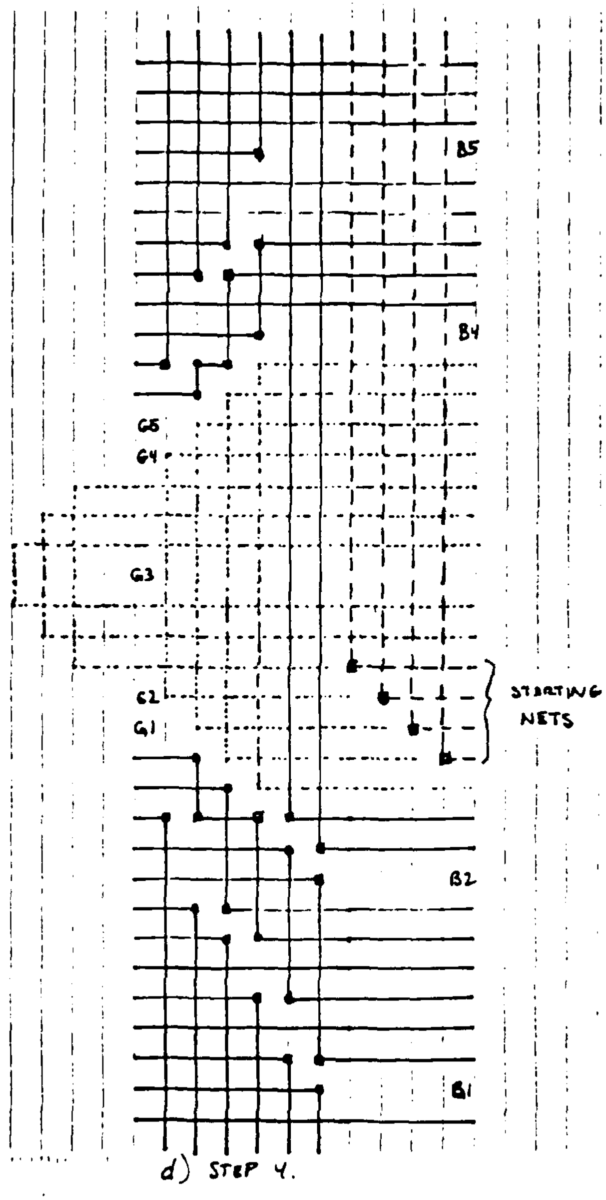
Let e_f and e_r denote the number of ending falling and rising nets, respectively, routed in Step 1. If $e_f > e_r$, the difference is balanced by routing $e_f - e_r$ starting falling nets. (If $e_f < e_r$, the symmetric routing is performed in the rising net portion of the channel, and if $e_f = e_r$, no balancing is needed.) If there are less than $e_f - e_r$ starting falling nets to be routed in the block, i.e., $s_f < e_f - e_r$, an additional $e_f - e_r - s_f$ vacant tracks remain in the falling portion. The pyramid is expanded into these empty tracks, and reshaped if necessary. Once again, the pyramid cannot be expanded into a track in the same column in which it was vacated (in Step 2). In order to separate the staircase pattern of the pyramid from the continuing net staircases, good tracks G2 and G4 are positioned just inside of G1 and G5, respectively (see Figure 3c). The centermost tracks vacated in Step 2 are not filled in, and become bad tracks B2 and B4, corresponding to good tracks G2 and G4, respectively. The pyramid and continuing net staircases are thus separated by using G2→B2 and G4→B4 as buffer tracks. As a result, the pyramid is expanded by e_f tracks in the first e_f columns at the top and bottom sides of the channel.



a) STEP 1. b) STEP 2.



c) STEP 3



d) STEP 4.

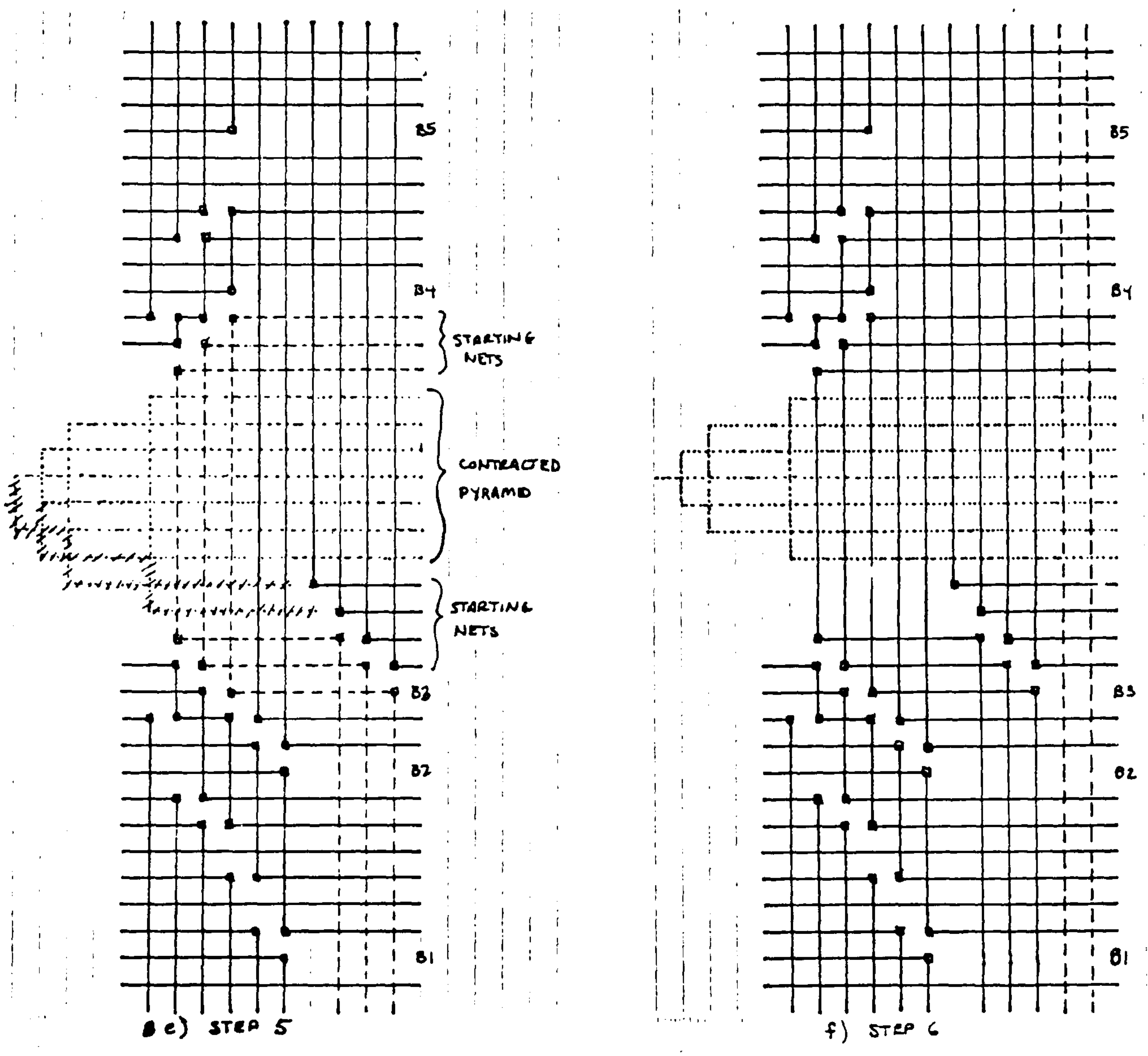


Figure 3. Phase 1.1 routing.

Step 4: Route one side's starting nets directly.

Let s_f and s_r denote the remaining number of starting falling and rising nets, respectively. If $s_f \geq s_r$, the lowest empty track of the pyramid is skipped (this will become B3), and the starting falling nets are routed directly into the next s_f tracks. The nets are routed into the s_f empty tracks in order from top to bottom, forming another staircase pattern (see Figure 3d). (The case $s_f < s_r$ is handled symmetrically.)

Step 5: Route remaining starting nets through the pyramid.

Next, assuming $s_f \geq s_r$, the remaining starting rising nets are routed using the outermost s_r paths of the pyramid. Good track G3, positioned in the center of the pyramid, separates the starting rising nets from the starting falling nets. Since $s_r \leq s_f$, there is always room to route all s_r nets into the pyramid, in staircase order. Note that if s_r is strictly less than s_f, some lower terminals do not contain nets. These $s_f - s_r$ empty terminals are put to the left of the rising nets routed in Step 5. This results in an extra $s_f - s_r$ tracks on one side of the pyramid which are not filled in. These $s_f - s_r$ tracks and good track G3 must be incorporated into the pyramid (see Figure 3e).

Step 6: Route vertical nets.

Finally, all ending and starting nets in block B have been routed; only vertical nets remain. These are routed trivially across the channel, as in Figure 3f. □

Notice that by routing ending nets before starting nets, the density of the permuted problem within each block is no greater than that of the original problem. Thus, Phase 1.1 uses at most d+5 tracks in all, d for routing and five empty good/bad tracks.

Phase 1.2 modifies the Phase 1.1 routing to move empty tracks via unit length vertical jogs (without changing layers). Such jogs do not interfere with the Phase 1.1 routing, but they do give rise to unit length segments of vertical overlap (see Figure 4). Phase 1.1 determines the relative track order of the nets in the current block, and Phase 1.2 does not affect this order. Recall that after routing a block, Phase 1.1 leaves five empty bad tracks scattered throughout the channel. In block B, Phase 1.2 moves the five empty bad tracks from the previous block, B−1, back into the middle of the channel to serve as the good tracks in the following block, B+1. These five tracks are in addition to the five good tracks being used by Phase 1.1 in block B.

Note that d+10 columns would be needed to jog a single empty track across all d+10 tracks. However, the positioning of all five empty tracks must be completed within the r columns of the block. In order to avoid long propagation of unit jogs, additional empty tracks, called ***reserved*** tracks, are distributed through the channel at intervals of r/5 tracks. These tracks (as well as the five empty tracks being moved in Phase 1.2) are invisible to Phase 1.1. The reserved tracks limit the propagation due to moving an empty track to at most r/5 columns, so no more than r columns are needed to move all five good/bad tracks to the correct positions.

The reserved tracks appear in exactly the same position at the beginning and end of the propagation of a bad track. Consider a group of $r/5-1$ tracks bounded on top and bottom by reserved tracks R_k and R_{k+1}, respectively (see Figure 4). If a bad track is to be propagated through this group, both R_k and R_{k+1} are filled in from above in the first column. After all $r/5-1$ tracks in the group jog, reserved track R_k is jogged, and becomes empty again. The groups in which the bad track starts and stops are similar, except they do not require the full $r/5$ columns to propagate.

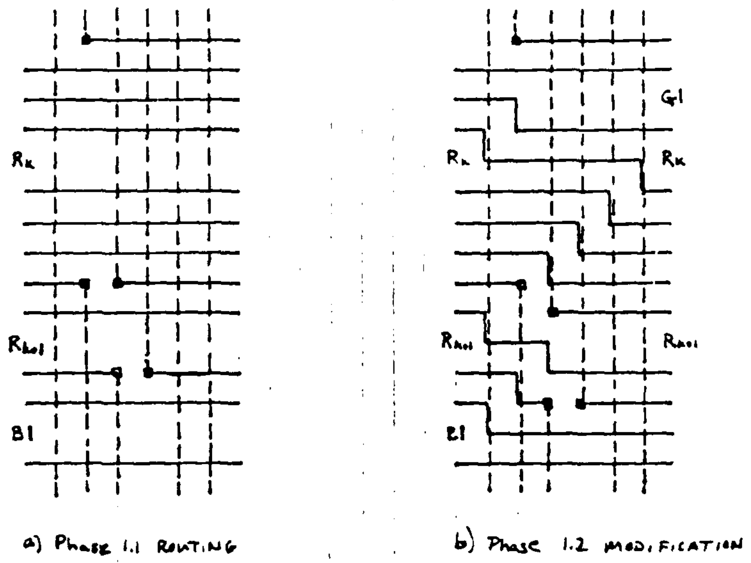


Figure 4. Propagating an empty track B1 to a good position G1. R_i denotes reserved track i.

That Phases 1.1 and 1.2 do not interfere with each other can be seen from the following argument. Take the $d+5$ track routing for block B, and insert five empty tracks corresponding to the five empty bad tracks from block $B-1$. Note that this has no effect on the Phase 1.1 routing, except to stretch the length of the vertical wires crossing these empty tracks. Next, insert reserved empty tracks between every $r/5-1$ tracks of the previous $d+10$. Now, to complete Phase 1.2, we need only to add jogs at appropriate places. A jog is one unit, into an adjacent empty track, so the relative order of the routing tracks is never changed. Consider a horizontal wire to be jogged. The original routing can only take two forms. Either it crosses a column horizontally or it bends at the column in which it jogs. Figure 5 enumerates these cases. Bends remain bends after a jog, while a crossing causes a unit vertical overlap.

Thus, the Phase 1.1 routing can be done by ignoring the Phase 1.2 empty tracks, and then Phase 1.2 can be added in by the method described, transforming a Manhattan routing into a unit vertical overlap routing.

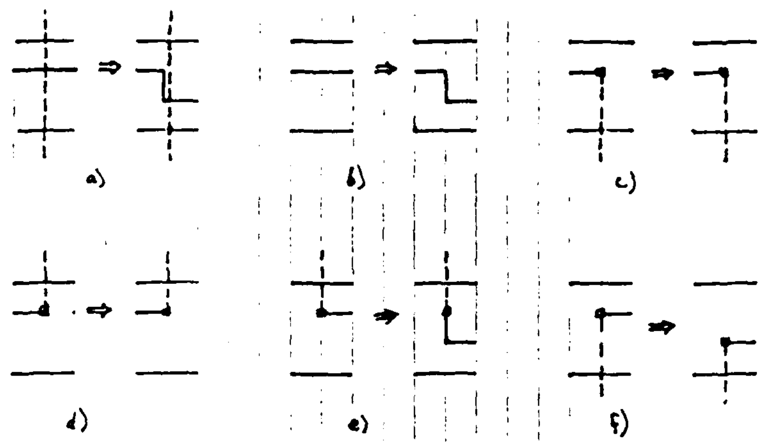


Figure 5. Adding unit jogs to Phase 1.1 routing.

Since reserved tracks are distributed at intervals of $r/5$ tracks through the remaining d+10 tracks, Phase 1.2 uses a total of $\frac{5(d+10)}{r}$ reserved tracks in addition to its five empty good/bad tracks.

Lemma 1: Phase 1 requires at most $(d+10) + \frac{5(d+10)}{r}$ tracks.

Since Phase 1 has routed all nets to the correct blocks, the densities of the top and bottom sections of the channel are each bounded by r. In Phase 2, we simply apply a no-overlap, knock-knee algorithm (e.g., [RBM81], [BB82]) which routes a channel in $2d-1$ tracks. Thus, the top and bottom sections can each be routed using $2r-1$ tracks. An additional two tracks are allowed for layer changes in connecting Phase 2 routing sections to the Phase 1 routing.

Lemma 2: Phase 2 requires at most $4r$ tracks.

Lemmas 1 and 2 yield the following theorem.

Theorem 1: Any two-terminal net channel can be routed in two layers using $d + O(\sqrt{d})$ tracks, allowing unit vertical overlap.

Proof: Phases 1 and 2 together route any two-terminal net channel routing problem. By Lemmas 1 and 2 we know that we have used $t = (d+10) + \frac{5(d+10)}{r} + 4r$ tracks. Choosing r in order to minimize t, we see that $r = \frac{\sqrt{5(d+10)}}{2}$, and so our algorithm uses $t = (d + 10) + 4\sqrt{5(d+10)}$ tracks. □

2.2. Extension to Multiterminal Nets

Next, we show how the algorithm can be extended to handle multiterminal nets. All top (bottom) terminals of a multiterminal net in a given block are collapsed to a single top (bottom) terminal in that block. The routing is essentially the same as for the two-terminal net algorithm, with a few additional cases. The strategy is to create two horizontal wires for each entering multiterminal net, one on the rising and one on the falling side of the channel. The upper and lower portions are treated as different nets, and each portion ends at its rightmost terminal.

Intermediate terminals are positioned within the block as if they were empty terminals. No vertical routing occurs in the falling (rising) net region in a column containing an empty lower (upper) terminal. Thus, an intermediate terminal positioned as if it were an empty terminal can be connected vertically to its horizontal portion, as in Figure 6.

The only difficulties arise in starting nets. Consider, for example, a net which begins at a lower terminal and has additional terminals in later blocks on both sides of the channel. The rising portion of such a net is routed just as a starting net in the two-terminal net algorithm, either straight across or backtracking through the pyramid. In addition, the lowest available routing track is used to begin the falling portion of the net. A net starting at the top terminal in the same column can then be routed in the same way without conflict. Four new horizontal wires begin in such a column (see Figure 6). However, for nets having all terminals after the starting terminal on the same side, only the necessary portion of the net is routed. Nets starting in block B which have terminals on both sides of B are routed as vertical nets. The lowest and highest available routing tracks are then used to begin the falling and rising portions of the net.

The number of tracks is potentially doubled by this method, yielding the following corollary to Theorem 1.

Corollary: Any multiterminal net channel routing problem can be routed in the (two-layer) unit vertical overlap model using $2d + O(\sqrt{d})$ tracks.

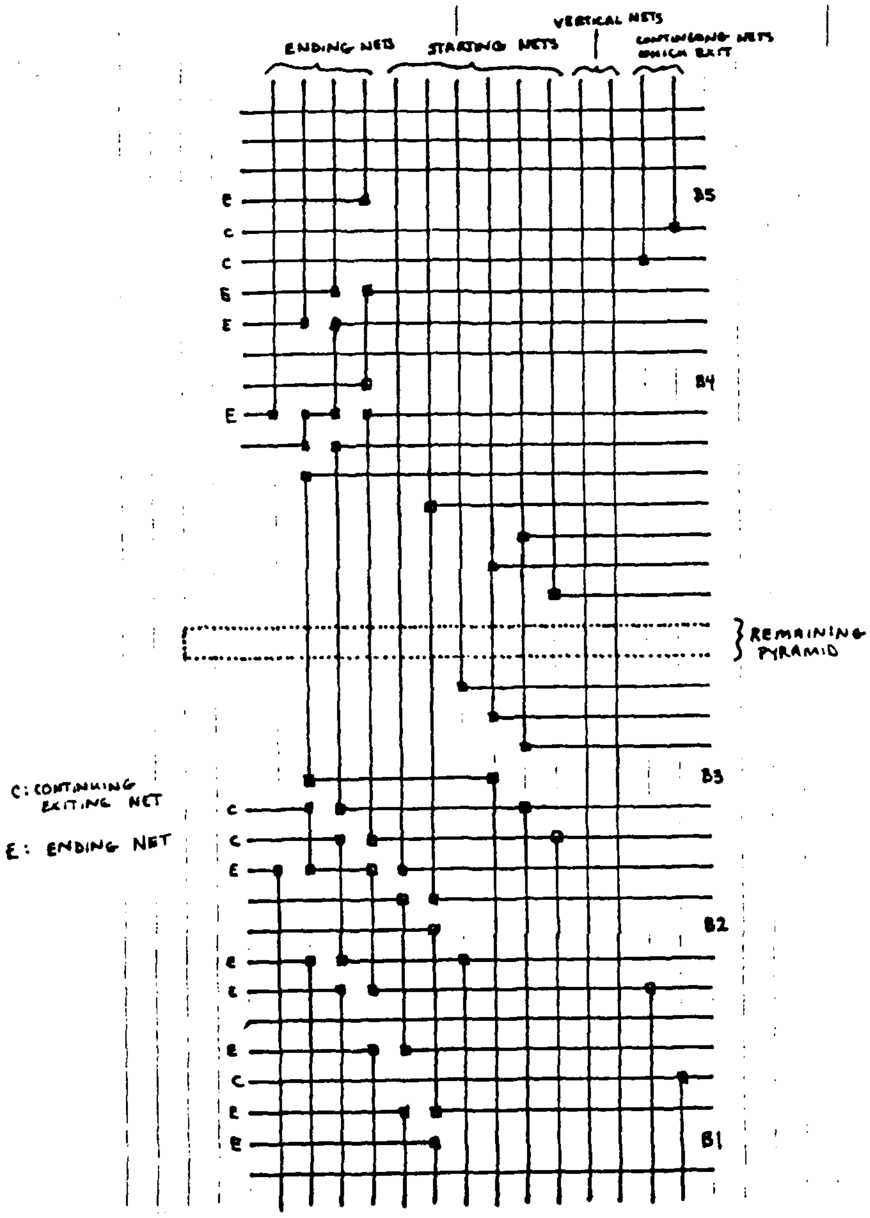


Figure 6. Phase 1.1 routing with multiterminal nets.

Notice that if the problem consists only of two-terminal nets and same-side multiterminal nets, the algorithm uses only d routing tracks, since only one horizontal wire is needed for each such net.

Recently, Gao and Kaufmann have improved the multiterminal net bound [GK87]. They transform a general multiterminal net problem of density d into an *extended simple channel routing problem (ESCRP)* of density $3d/2 + O(\sqrt{d \log d})$. They then modify our two-terminal net $d + O(\sqrt{d})$ algorithm to route the ESCRP, yielding a $3d/2 + O(\sqrt{d \log d})$ multiterminal net algorithm.

3. Applying the Strategy to Other Routing Models

In this section we shall show that our routing strategy can be generalized to cover other routing models. The robustness of our method is appealing for two reasons. First, an algorithm which is tolerant to model variations is more likely to be useful in practice. Second, the strategy can lead to new models which are amenable to improved channel routing solutions (the unit vertical overlap model is such an example).

3.1. The Manhattan Model

Not surprisingly, our strategy covers the Manhattan model, since it is based on the Manhattan routing algorithm of [BBL84]. Empty columns, rather than extra empty tracks are used to separate the staircase patterns. Three empty columns are positioned per block for this purpose. Thus, Phase 1.2 is no longer required. Instead, $O(f)$ tracks are used at the top and bottom of the channel to move empty columns into the correct positions where f is the flux, and the optimal block size becomes $O(f)$, which yields a $d + O(f)$ upper bound for two-point nets (see the example in Figure 7) and a $2d + O(f)$ upper bound for multipoint nets.

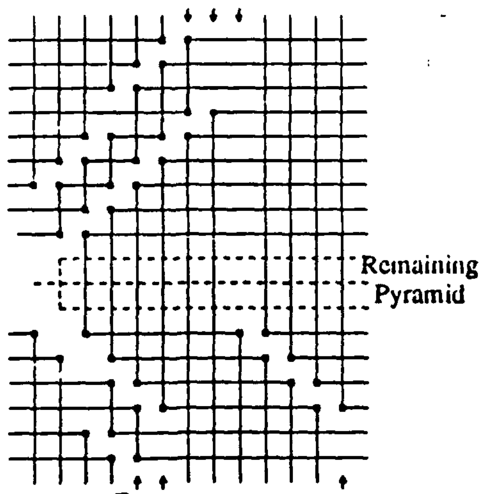
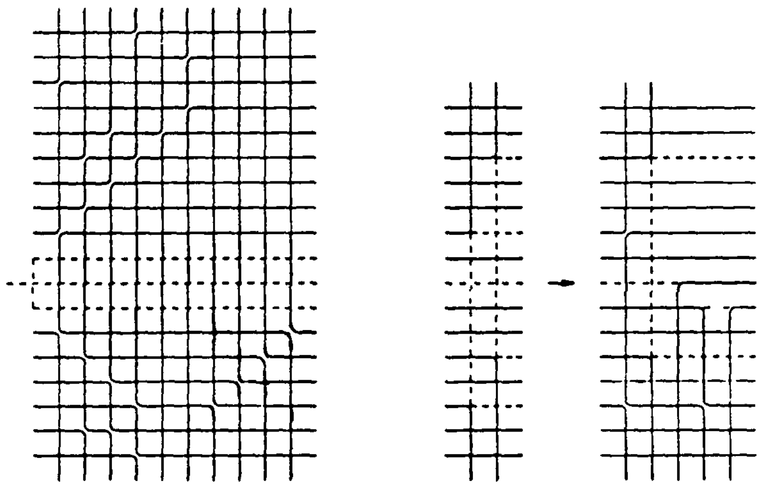


Figure 7. Applying the strategy to the Manhattan model.

3.2. The Knock-Knee Model

It is a bit surprising, however, that the routing strategy described in Section 2 gives a $2d-1$ track algorithm for two-layer knock-knee routing of two-point net problems as well. This reproduces the results of [RBM81] and [BB82]. To route in the knock-knee model, the block size is chosen to be one, and the staircases are not pulled apart, but instead are allowed to knock-knee. Thus, Phase 2 is unnecessary, as nets are routed exactly into the correct columns. The routing uses d tracks, and just as in [RBM81], [BB82]. An additional $d-1$ empty tracks must be included, one between each pair of routing tracks, to allow for layer changes (see the example in Figure 8a).



(a) Two-layer routing. (b) Three-layer modification.

Figure 8. Applying the strategy to the knock-knee model.

We do not know whether or not the knock-knee routings produced by this algorithm can be wired in three layers without adding any extra tracks. However, for problems which do not contain any same-side nets, with a small modification, the strategy can be shown to produce d-track knock-knee routings which are three-layer wirable. This matches the optimal results of [PL84], who also route only top-to-bottom nets. First, note that filling in vacated tracks with continuing nets is not really necessary. In the original algorithm, this was done to maintain the pyramid in the center of the channel. Instead, the pyramid can be scattered among the rising and falling nets. Each backtracking path of the pyramid consists of two tracks. We need only to ensure that for each path, one of these two tracks is on the rising side and that the other is on the falling side of the channel. This invariant is maintained across columns which contain two terminals: two beginning nets fill in one backtracking path of the pyramid, two ending nets create one new backtracking path, and one net of each type vacate and fill in the same track, without affecting the pyramid.

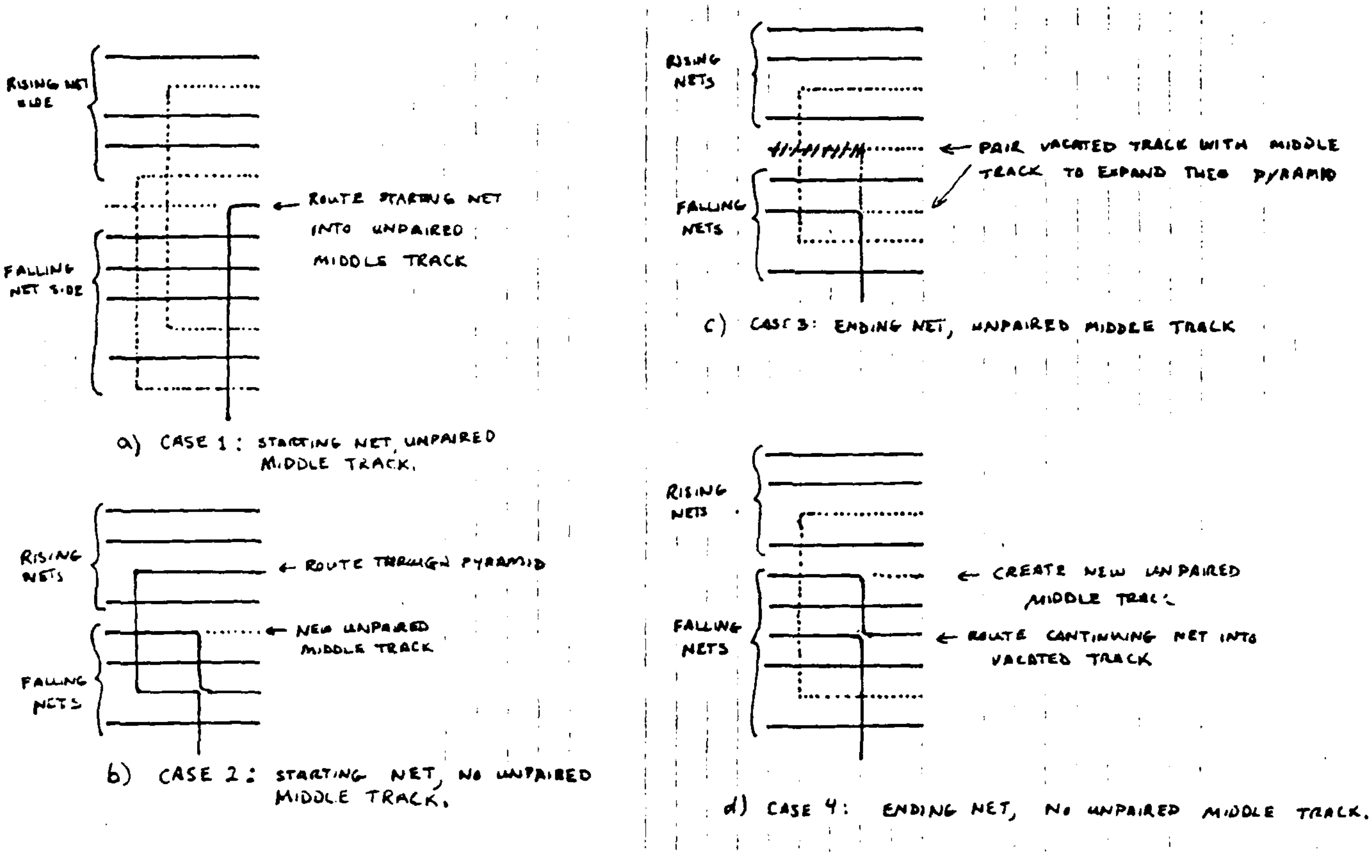


Figure 9. Routing single terminal columns.

Recall that in a column having precisely one empty terminal, the pyramid must be balanced. Empty tracks between the rising and falling portion can be assigned to either the rising or falling side, in order to maintain balance as before. The only problem is when there are no empty middle tracks. In this case, we create an unpaired middle track by routing a single continuing net into the side with the excess empty track, leaving an empty track in the middle. The track becomes the center of the pyramid, and is not used for routing until another single terminal column is routed (see Figure 8b). There are four single-terminal cases to be considered, shown in Figure 9. First, consider starting nets. A starting net is routed into the unpaired middle track, if it exists (Figure 9a). If not, the net is routed through the pyramid, and an unpaired middle track is created by routing a continuing net into the empty track of the pyramid backtracking path used (Figure 9b). Otherwise, the net is ending. If there is an unpaired middle track, then it is paired with the track being vacated to expand the pyramid by one path (Figure 9c). As before, if there is no unpaired middle track, one is created by routing a continuing net, this time into the track vacated by the ending net (Figure 9d).

The advantage of this modified technique is that it can be shown to produce layouts which are three-layer wirable. Observe that a column of the layout may contain no more than two knock-knees. Further, among columns which contain exactly two knock-knees, the lower ones are all of the same type (west to south and north to east bends), and the upper ones are of the opposite type. Preparata and Lipski [PL84] have shown that a layout with these characteristics can always be wired in three layers, without adding any extra tracks.

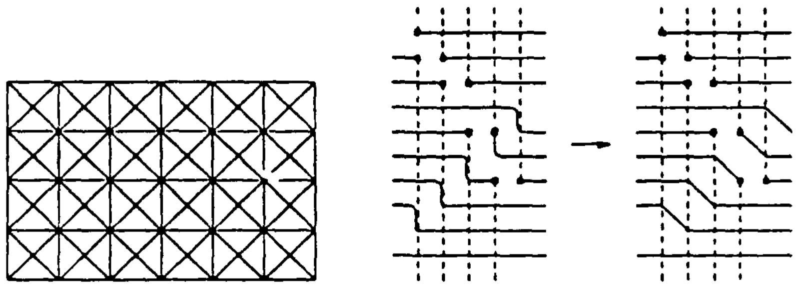
3.3. 45-Degree Diagonal Wires

Finally, consider augmenting the rectilinear grid by allowing *diagonal* wires at 45 degrees from the horizontal and vertical directions, as in Figure 10a. Diagonal segments connect gridpoints of the original grid. Notice, however, that adjacent parallel diagonal segments are separated by only $\sqrt{2}/2$ units. Thus, in order to realize reasonable layouts in this model, different wires are not allowed to be simultaneously routed in two adjacent parallel diagonal segments in the same layer.

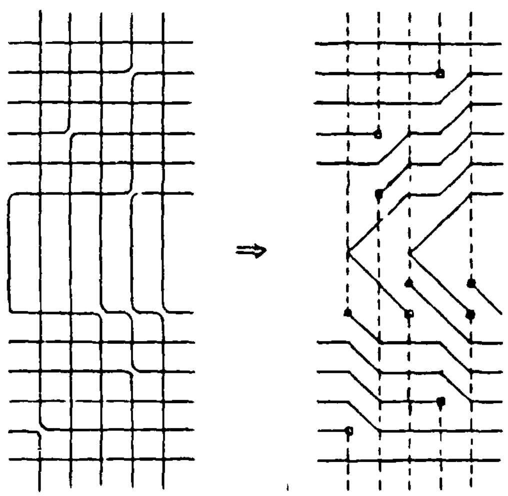
The $d + O(\sqrt{d})$ unit vertical overlap algorithm can be applied directly to the diagonal grid without using any overlap. Unit vertical overlaps, due to unit jogs of horizontal wires, are replaced by diagonal segments, as in Figure 10b. It can be shown that some CRPs require at least d tracks in this model.

We also note that if *adjacent* diagonal wires are allowed, it is not difficult to modify the algorithm so that d tracks suffice for all CRPs. A starting net can be jogged immediately into a vacated track, in a single column, so the block size can be chosen $r=1$. The algorithm is similar to the $2d-1$ knock-knee algorithm described in the previous section. The transformation from such a knock-knee routing to this diagonal model is shown in Figure 10c. Rather than filling in an exiting net with the centermost continuing net directly, nets are jogged by one to fill in the empty track using diagonal segments, in a single column. Backtracking is done using diagonal wires in layer

2, and therefore do not interfere with the vertical wires in the center of the channel. The *d*-track algorithm reproduces the results of Sarrafzadeh [Sa87].



(a) Diagonal grid. (b) Transforming unit vertical overlaps to diagonals.



c) TRANSFORMING KNOCK-KNEE ROUTING TO ADJACENT DIAGONALS.

Figure 10. Routing using diagonal wires.

Recently, Lodi, Luccio, and Pagli [LLP87] have investigated a different diagonal wire routing model. Their model contains only 45-degree diagonal paths, i.e., our diagonal grid without the vertical and horizontal lines. They give an algorithm for this model which is optimal for instances containing only short nets.

4. The General Algorithm for L Layers

4.1. Two-Terminal Nets

In this section we shall extend the two-layer unit vertical overlap routing algorithm to multilayer channel routing. The channel is defined to contain L layers, and wires are allowed unrestricted overlap. Obviously, at least d/L tracks are required to route any CRP in this model. Furthermore, Hambrusch [Ha83] proved an existential lower bound of $\frac{d}{L-1}$, extending the two-layer result of Leighton [Le82]. This lower bound is not surprising. Consider a stretch of the channel in which at least one track contains L nets. This track is as an obstacle to any net which attempts to cross the channel; even trivial nets must be routed around this area.

Surprisingly, our two-terminal net strategy comes very close to this lower bound. Again, Phase 1 uses a rising/falling strategy with empty tracks maintained in the middle of the channel. In the falling net portion of the channel, each horizontal routing track contains up to $L-1$ different nets, in layers 2 through L. Only layer 1 is reserved for vertical connections. The organization of the rising nets is slightly different. Nets are routed horizontally in layers 1 through $L-1$, and layer L is used to route vertical wires. The advantage of this assignment is that a starting rising net and a starting falling net can be routed in the same column without conflicting use of the same vertical layer. Thus, the backtracking pyramid is not necessary.

In this model, both a track and a layer are required to uniquely specify a net's horizontal position; each track is said to consist of $L-1$ *lanes*. Clearly, in a track containing $L-1$ falling nets, only the net in layer 2 has access to the vertical layer. But when the net in layer 2 exits and vacates the lane, the net in layer 3 may access the vertical layer, and so forth. All falling (rising) ending nets which are packed into the lowest (highest) horizontal layers are defined as ***positioned***, while a falling (rising) ending net whose access to layer 1 (L) is blocked by a continuing net in a lower (higher) layer is ***nonpositioned***.

Unfortunately, we are not able to position and then exit all ending nets within a single block. Instead, some ending nets do not exit in their target block, and instead continue to the right to exit in a later block. We distinguish these "late to exit" nets from those which have just become ending in the current block, B. Nets which are supposed to exit in B are called ***new*** ending nets, while those which became ending in an earlier block but were unable to exit are called *extended* nets. A block may contain up to $4r$ falling and $4r$ rising extended nets, so the density of the remaining Phase 2 subproblems is $O(r)$.

Phase 1 still proceeds block-by-block from left to right. As before, Phase 1 has two parts, Phases 1.1 and 1.2. In Phase 1.1, starting and vertical nets are always routed in the correct block. All remaining terminals are available to route ending nets. As before, positioned ending nets are exited in the leftmost columns, starting nets are routed next, followed by vertical nets. Falling (rising) ending nets are exited in order from the highest to the lowest (lowest to highest) tracks. The highest (lowest) continuing nets, i.e., those nearest the empty middle tracks, are then routed into the vacancies. However, starting falling and rising nets are routed directly across the middle empty tracks, using layers 1 and L, so no backtracking pyramid is needed. Consequently, good tracks G2, G3, and G4 are no longer necessary.

Phase 1.2 performs two functions in the multilayer algorithm. Phase 1.2a moves two bad tracks (B1 and B5) from the previous block into good track positions for the next block, using the leftmost columns of the block. Phase 1.2b positions ending nets, so that they may exit in a later block, using the remaining columns. Bad track B1 (B5) is not one completely empty track, but rather as a set of $L-1$ empty lanes distributed through the falling (rising) region. These empty lanes will be moved back into the middle of the channel, one lane at a time, by jogging only layer L (layer 1) in Phase 1.2a. As a result, positioned falling (rising) nets in layer L (layer 1) may become unpositioned during this process. Thus, these nets are called *semi-positioned*, while the nets positioned in layers 2 through $L-2$ are *fully* positioned. In the Phase 1.2a columns, only fully positioned nets are feasible for exiting in Phase 1.1, while in the Phase 1.2b columns, both semi- and fully positioned nets are feasible.

The following invariants are maintained by the algorithm.

Invariant I1: All extended ending nets in a block are positioned when they enter that block.

Invariant I2: At most $L-1$ falling and $L-1$ rising fully positioned nets become unpositioned in a block.

Invariant I3: At most $4r$ falling and $4r$ rising extended nets enter any block.

Phase 1.1 is now outlined in detail (refer to Figure 11 for examples). Where the description is limited to the falling nets, the rising net case is symmetric. Let s_f (s_r) be the number of starting falling (rising) nets and v be the number of vertical nets in the current block, B. The rightmost v top and bottom terminals are reserved to route vertical nets, and the next s_f bottom (s_r top) terminals route starting nets. The remaining $r - s_f - v$ ($r - s_r - v$) terminals are available to route ending nets. For simplicity, we will assume that all nets are top-to-bottom nets. The extension to handle same-side-nets is straightforward.

Step 1: Exit positioned ending nets.

As many feasible positioned falling ending nets as possible are exited in the leftmost columns of the block, i.e., until either the feasible positioned falling ending nets run out or $r - s_f - v$ ending falling nets are routed. The falling nets are exited in order, from the highest track to the lowest. Recall that a track may contain more than one positioned net. In this case, all feasible positioned nets in a track are exited in order before proceeding to the next track (see Figure 11a).

Step 2: Fill in vacated lanes with continuing nets.

In the next step, the lanes vacated by falling nets in Step 1 are filled in with continuing nets from the top of the group of falling nets. Notice that a lane cannot be refilled in the same column in which it is vacated. In the worst case, all $L-1$ nets in a track exit, and the continuing nets may not begin to fill in the track until $L-1$ columns after the first net exits. Thus, there is an $L-1$ column lag between the vacating and refilling of a lane. Continuing nets are not routed until $L-1$ empty lanes have been vacated in the interior of the falling net region, and one continuing net is routed in each successive column in which *both* an exiting falling and rising net is routed (see Figure 11b). Notice that in routing the continuing nets, a nonpositioned ending net may be uncovered. The net effectively becomes the highest falling positioned net, and as such, is immediately exited in the next column (see Figure 11b, col. 5). (Of course, in an actual implementation, Steps 1 and 2 must run together, in order to integrate ending nets uncovered in Step 2 into the falling net routing.)

Step 3: Balance the rising and falling ending net columns.

Let e_f and e_r denote the number of falling and rising nets which exit in the current block in Step 1. If $e_f > e_r$, then $e_f - e_r$ starting falling nets are routed directly into the lanes vacated by falling ending nets, rather than first filling in these exiting nets with continuing nets as in the two-layer algorithm (see Figure 11c). This is so that good track G4 is not needed. The starting net must change from layer L in the rising portion to layer 1 in the falling portion. This is accomplished using a via in an empty middle track.

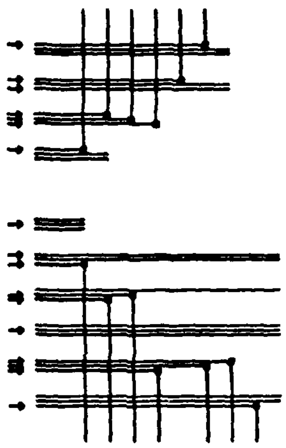
If there are not enough starting falling nets to balance, i.e., $s_f < e_f - e_r$ when $e_f > e_r$, then starting nets are routed in the first s_f columns, and the highest falling continuing nets are routed into the vacated tracks in the remaining $s_f - (e_f - e_r)$ columns. Again, if this process uncovers an ending net, it becomes the highest falling positioned net, and is exited immediately (or remains positioned into block $B+1$, if it is uncovered in the last column of block B). Routing for the case in which $e_f < e_r$ is symmetric. If $e_f = e_r$, no nets are routed in this step.

Step 4: Route starting rising and falling nets.

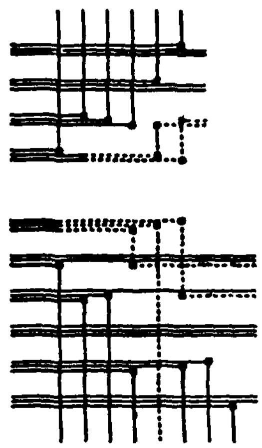
In the remaining columns, no more feasible positioned exiting nets remain to be routed in the current block. Starting rising and falling nets are routed next. The starting nets reclaim some of the empty lanes in the middle of the channel, thus expanding the groups of falling and rising nets, which had contracted during the routing of the exiting nets. The starting rising (falling) nets are routed in layer 1 (layer L) and fill in tracks in order from layer 1 up to layer $L-1$ (layer L down to layer 2) (see Figure 11d). Two empty tracks between the rising and falling nets must always exist to allow the two nets to change layers for entry into partially filled tracks.

Step 5: Route vertical nets across the channel.

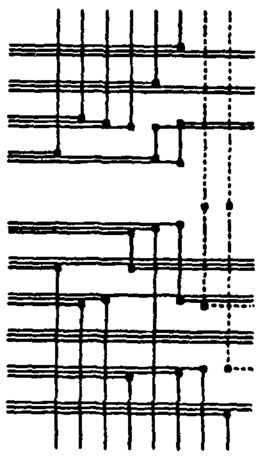
Finally, in the remaining columns, vertical nets are routed trivially across the channel. They are routed through the falling nets in layer 1, through the rising nets in layer L, and joined by a via using one of the two middle empty tracks (see Figure 11e).



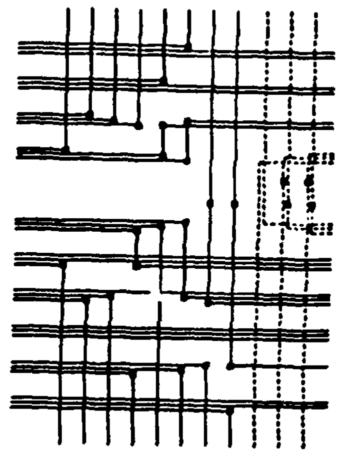
(a) Step 1. Exit positioned ending nets (arrows indicate ending nets).



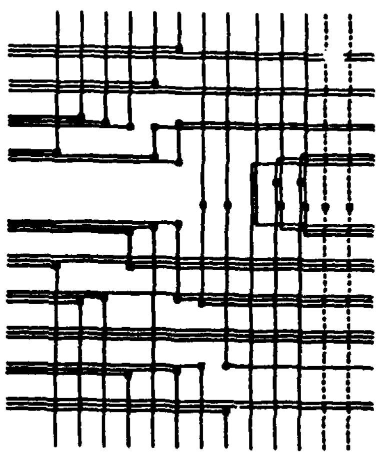
(b) Step 2. Fill in with continuing nets.



(c) Step 3. Balance with entering nets.



(d) Step 4. Route starting rising and falling nets.



(e) Step 5. Route vertical nets.

Figure 11. Phase 1.1 routing for two-point net multilayer algorithm.

Phase 1.2 has two purposes in the two-terminal net algorithm. In Phase 1.2a, the $L-1$ empty bad lanes in the falling and rising regions from the previous block, $B-1$, are moved into the empty middle tracks for use in the next block, $B+1$. Next, all ending nets which do not exit in B are positioned, in Phase 1.2b. The reserved tracks are evenly spaced $y = \frac{r-(L-2)}{3(L-1)}$ tracks apart in order to achieve both of these requirements.

The first $L-2 + (L-1)y$ columns of a block B are used to move the bad lanes into the empty tracks in the center of the routing. In Section 2, we showed how to propagate an empty track using y columns. In the L-layer algorithm, a bad "track" is actually a collection of $L-1$ empty lanes, but these lanes may be spread among many different tracks. The empty lanes in the falling (rising) region must first be packed into the highest (lowest) layers. This is handled as shown in Figure 12a, by shifting nets downward (upward) one lane per column. The bad lanes are then jogged one at a time through layer L (layer 1) into the middle of the channel (see Figure 12a). Vias at the beginning and end of a run can connect the layer L (layer 1) jog with the proper beginning and ending layers.

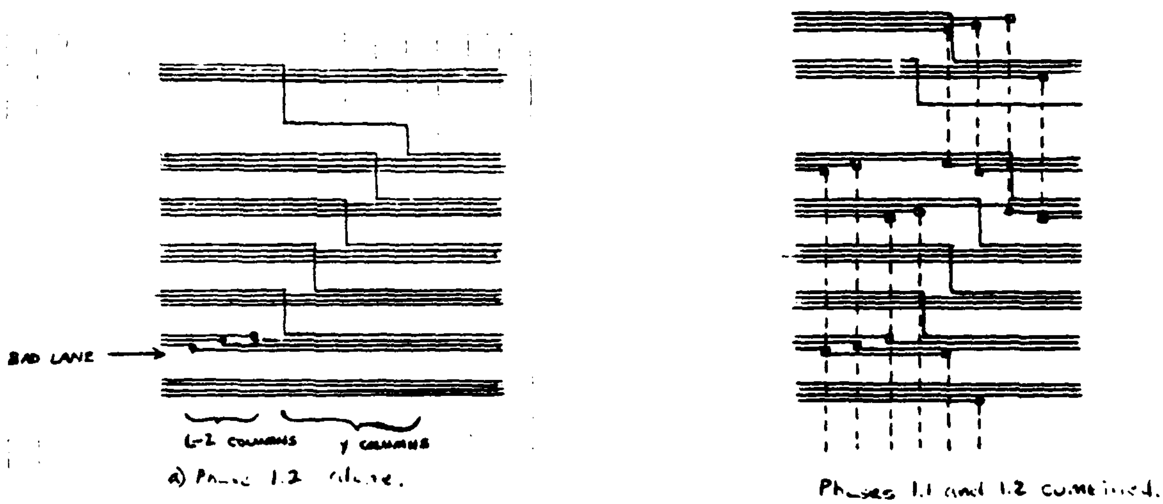


Figure 12. Phase 1.2a routing (falling net region).

Phase 1.2a does not conflict with Phase 1.1. Except at the beginning and end of a run, Phase 1.2a uses only layer L (layer 1) in the falling (rising) region, while Phase 1.1 exits only fully positioned ending nets, in layers 2 through $L-1$. At the beginning and end of a run, the highest filled lane may move into layer L, while the lowest filled lane moves into layer 1, without conflict, as in Figure 12b.

The remaining $2(L-1)y$ columns are used in Phase 1.2b to position ending nets. This is done by propagating an extra empty track through the entire width of the channel using the reserved tracks. During this process, the nets in a jogged track can change layers, as shown in Figure 13. A set of $L-1$ consecutive columns are used to position a track. We describe the procedure for the falling net case. Let x be the number of ending nets in the track. First, position the ending nets using the first x columns. In the i^{th} column, the i^{th} highest ending net is first jogged and then moved down into layer $i+1$. Thus the relative order of the ending nets is unchanged. Next, move the remaining $L-1-x$ nets into layers $x+2$ through L. This is accomplished in a manner similar to the first step. In the $(x+i)^{th}$ column, the i^{th} highest non-ending net is first moved up into layer $L+1-i$, and then jogged down. The relative order of the non-ending nets is also unchanged.

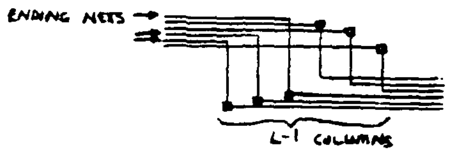


Figure 13. Positioning a track in Phase 1.2b.

Each track has a chance to jog and position its nets within $(L-1)y$ columns. However, no positioning is done in a track if it would conflict with nets exiting or entering the track in Phase 1.1 (Steps 1 and 2). Instead, the track is simply jogged without rearrangement. To ensure that every track has a chance to position its ending nets, the entire $(L-1)y$ column process is performed twice. Any track unable to be positioned in the first pass will certainly be positioned in the second, since the opportunity to exit nets in Phase 1.1 occurs only once. Thus, all tracks are positioned in Phase 1.2b, and there is no conflict with Phase 1.1. Phase 1.2b uses $2(L-1)y$ columns.

Now we show that the invariants are maintained by the Phase 1 routing procedure.

Invariant I1:

A track which is exited and refilled (Phase 1.1) before the second sweep of Phase 1.2b will be positioned in the second sweep, and remain so to the end of the block. If the track's turn to exit is during the second sweep, then the track was positioned during the first sweep. Thus, its ending nets are already positioned and therefore exit. The last track which exits nets in a block may not be able to exit all of its positioned nets. But such a *partially exited* track is not refilled in Step 2, so its remaining nets do not become unpositioned.

Invariant I2:

No nets become unpositioned during Phase 1.1. Phase 1.1 exits all positioned nets from a track before attempting to fill it in with continuing nets. (The last track from which positioned nets exit may be partially exited, but this track is not refilled in the current block.) Only continuing nets are used to fill in, since any ending nets which are uncovered are exited immediately.

In Phase 1.2a, at most $L-1$ fully positioned rising and falling nets become unpositioned. In the first $L-2$ columns, no positioned nets are moved. Next, the $L-1$ bad lanes in the falling (rising) region are jogged into the middle of the channel using only layer L (layer 1), which by definition contains no fully positioned nets. The only exception is at the end of a run. The highest falling (lowest rising) track may contribute a fully positioned net to complete each of the $L-1$ bad lane jogs.

Finally, Phase 1.2b obviously does not unposition any nets, so I2 is satisfied.

Invariant I3:

Next, we ensure that if no more than $4r$ falling (rising) extended nets enter a block, B, then no more than $4r$ extended nets enter $B+1$. We argue for the falling nets; The rising net argument is similar. Let x_f be the number of falling extended nets which enter B, and e_f be the number of *new* falling ending nets in B. By definition, e_f lower terminals are available for routing falling ending nets, so we need only ensure that enough ending nets are fully positioned. By I1, all x_f extended nets are either fully positioned or semi-positioned at the start of the block. At most $\frac{x_f}{L-1}$ of these are semi-positioned, since all other nets in the track of a semi-positioned net are fully positioned. By I2, at most $L-1$ fully positioned nets become unpositioned. Thus, at least $\min\{\frac{(L-2)x_f}{L-1} - (L-1), e_f\}$ nets exit in B. If $e_f \leq \frac{(L-2)x_f}{L-1} - (L-1)$, we are done, so assume that $e_f > \frac{(L-2)x_f}{L-1} - (L-1)$. Then the number of extended nets becomes $x_f + e_f - \frac{(L-2)x_f}{L-1} + (L-1) = \frac{x_f}{L-1} + e_f + (L-1)$. Since $x_f \le 4r$, $e_f \le r$, and $L \ge 3$, this quantity is at most $3r + (L-1)$. Further, r will be chosen to be at least $L-1$, therefore the number of extended nets is at most $4r$.

Lemma 3: Phase 1.1 of the L-layer, two-terminal net algorithm uses at most $\frac{x_f}{L-1} + O(r/L)$ tracks for $r \ge L$.

Proof: At most d continuing nets and $8r$ extended nets cross any block boundary (by I3). Since ending nets are routed before starting nets within a block, the maximum number of nets crossing any column is no more than $d + 8r$. Each net requires one lane over the set of blocks which it spans, and there are $L-1$ horizontal routing lanes per track. Two empty tracks are necessary in between the rising and falling nets for the vias placed in Step 4. An additional two tracks are included to account for the possibility that the number of rising nets and the number of falling nets are not multiples of $L-1$, and therefore the highest falling and lowest rising nets' tracks are only partially filled. Finally, $L-1$ empty bad lanes, the equivalent of one track, remain in the rising and in the falling nets after routing exiting nets. Thus, at most $\left\lceil \frac{d+8r}{L-1} \right\rceil + 6$, or $\frac{d}{L-1} + O(r/L)$ tracks are used in Phase 1.1. □

Lemma 4: Phase 1.2 of the L-layer, two-terminal net algorithm requires at most $O((d/r) + 1)$ tracks.

Proof: Phase 1.2 spaces empty tracks $\frac{r-(L-2)}{3(L-1)}$ tracks apart among the $\frac{d}{L-1} + O(r/L)$ routing tracks. Only three additional empty tracks are required: the two bad tracks from the previous block plus one empty track to propagate back and forth across the channel to position ending nets. Thus, assuming $r \ge 7(L-1)$, at most $O\left(\frac{d+r}{r}\right) = O((d/r) + 1)$ tracks are used. □

The top and bottom sections each have density at most $4r$. Phase 2 employs the simple multilayer algorithm of [BB86], which uses $\frac{d}{\lfloor L/2 \rfloor - 1}$ tracks for $L \ge 3$.

Lemma 5: Phase 2 of the L-layer, two-terminal net algorithm uses at most $O(r/L)$ tracks.

Proof: Simply substitute $4r$ for d. □

Phases 1 and 2 together route an L-layer channel in a total of $\frac{d}{L-1} + O(r/L) + O((d/r) + 1)$ tracks. The block size, r, chosen to minimize this quantity is $r = O(\sqrt{dL} + L)$, which, along with the results of Section 2, implies the following theorem.

Theorem 2: Any two-terminal net channel can be routed in the L-layer arbitrary overlap model using $\frac{d}{L-1} + O(\sqrt{d/L} + 1)$ tracks, for $L \geq 2$.

4.2 Multiterminal Nets

Finally, we generalize the multilayer algorithm to multiterminal nets. The strategy used in our two-layer algorithm applied to L-layer routing would use $2d$ horizontal routing lanes and thus lead to a $\frac{2d}{L-1} + O(\sqrt{d/L} + 1)$ track algorithm. But this is inferior to even the simplest multiterminal net routing algorithm of [BB86]. However, if we reserve two layers, 1 and L, for vertical routing, then we can route with only $d + O(r)$ horizontal routing lanes, in layers 2 through $L-1$.

In Phase 1, all top (bottom) terminals of a net in a given block are collapsed to a single top (bottom) terminal in that block. A net is classified as rising (falling) in block B if the net's next terminal in a block $\geq B$ is on the top (bottom) of the channel. A net which has both an upper and lower terminal in the next block containing the net is *nonoriented*. As before, all top connections are made using layer L and all bottom connections use layer 1. However, layers 1 and L are both reserved for vertical wires in *both* the falling and rising portions of the channel. This allows continuing nets which change orientation (i.e., from falling to rising or vice-versa) to exit and to move into the opposite side of the channel, all in the same column.

Our multiterminal net strategy is largely the same as the one for two-terminal nets, with some modifications to accommodate continuing exiting nets and nonoriented nets. We first describe the changes to Phase 1.1. An example of Phase 1.1 multiterminal net routing is shown in Figure 14.

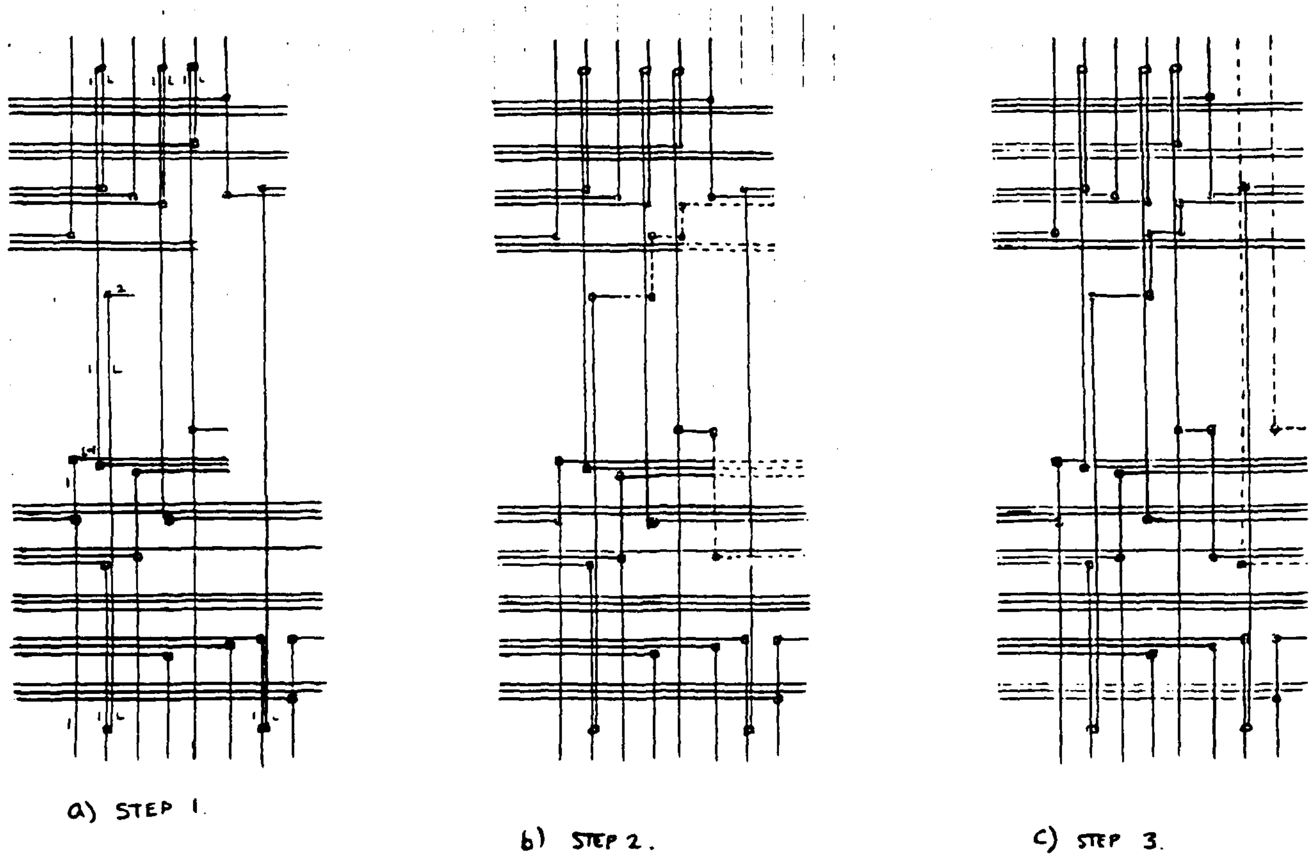
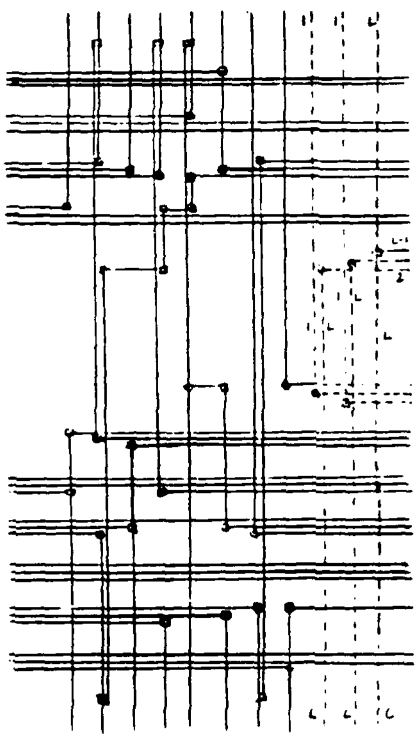
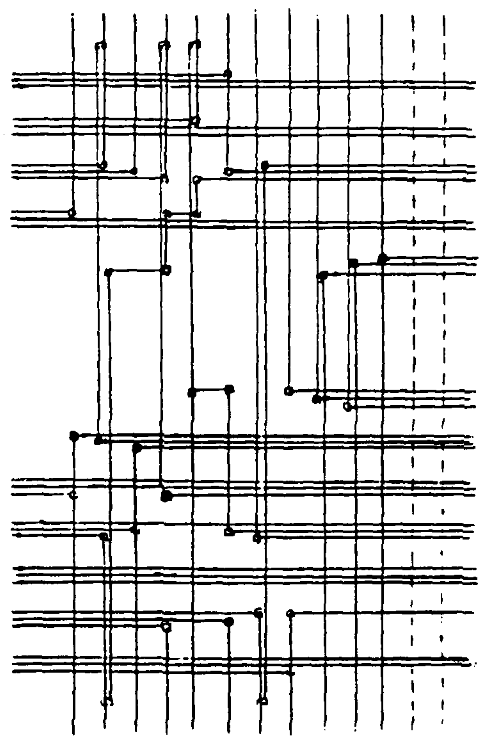


Figure 14. Phase 1.1 multilayer routing with multiterminal nets.



d) STEP 4.



e) STEP 5.

Continuing exiting nets are positioned and exited along with the ending nets, in Step 1. Exiting falling (rising) nets, both continuing and ending, are positioned in the lowest (highest) horizontal layers. A falling (rising) net positioned in layer $L-1$ (layer 2) is now semi-positioned, since layer L (layer 1) routes no horizontal wires. Note that a continuing exiting net positioned, for example, in layer 2 does not vacate its lane when it exits. Such a net must be moved into a different track upon exiting to allow the net in lane 3 to exit. Further, a continuing net which exits may change from rising to falling (falling to rising), and must therefore be moved into the opposite region. We maintain two empty tracks for layer changes in Phase 1.1, the highest and lowest tracks of the Phase 1 region.

In Phase 1.1, Step 1 is modified to move continuing nets into a different track in the same column in which they exit, as follows.

Step 1: Exit positioned ending nets (both ending and continuing).

As many feasible positioned ending nets as possible, both ending and continuing, are exited in the leftmost columns of the block. The falling (rising) nets are exited in order, from the highest track to the lowest (lowest to highest). Ending nets are exited as in the two-point net algorithm; continuing nets which exit and nonoriented nets are handled as follows.

First, consider a falling continuing net which exits. If the net remains falling or becomes nonoriented, the net is routed up into the empty lane in the falling region which was vacated $L-2$ columns earlier, using layer 1. This is the lane that would have been filled in from above in Step 2 or 3. In the first $L-2$ columns of the block, the net is routed into the lowest available lane in the middle of the channel (see Figure 14a, column 1).

If the falling continuing net changes orientation, it exits using layer 1, and then in track 1 it is moved into layer L and routed back across the channel into the rising region. If a rising net exits in this column and leaves a vacant lane to be filled in by Step 2, a (different) vacant lane is normally refilled in this column, using the centermost continuing net. However, in this case, the vacant lane is instead filled in by the falling-turned-rising net (a rising-turned-falling example is shown in Figure 14a, column 3). Otherwise, if no continuing net would have been routed in Step 2 to fill in a vacant lane, the falling-turned-rising net is simply routed into the highest available lane in the middle of the channel (see Figure 14a, column 2).

Finally, if a nonoriented net exits, e.g., from the falling region, its top connection is routed in the same column. The net exits as normal in the falling region, changes into layer L in track 1, and then is routed back across the channel. If the net is continuing, it is also placed in a new lane in the appropriate region, as described above (see Figure 14a, column 5).

Step 2 is essentially the same as in the two-point net algorithm, except that no net is routed in columns which are filled in by Step 1. Similarly, in Step 3, starting nets are not routed into a vacant lane filled in by Step 1. Instead, the starting net is routed into the lowest available falling (highest available rising) lane. Step 4 must be modified to handle starting nets which have a terminal on both sides of the channel in the current block (as well as terminals in later blocks). These nets are routed after all one-side starting nets have been routed. Both terminals are routed in the same column, and the net is placed in the next available lane in the appropriate region (see Figure 14d, columns 9-11). Finally, Step 5 routes vertical nets straight across the channel, in layer 1 (there is no longer any need to change layers).

Phase 1.2 is essentially the same as in the two-terminal net case. Phase 1.2a moves $L-2$, rather than $L-1$, empty lanes in the falling and rising regions into the middle of the channel. Again, this is done by jogging semi-positioned nets, in layers $L-1$ and 2, respectively. As before, since Phase 1.1 does not exit semi-positioned nets in the Phase 1.2a columns, there is no conflict with Phase 1.1. Phase 1.2a uses the first $(L-3) + (L-2)y$ columns of the block.

The remaining $2(L-2)y$ columns are used by Phase 1.2b to position exiting nets, both ending and continuing. The same two pass procedure is used for the multiterminal net algorithm.

The multiterminal net algorithm maintains the following three invariants.

Invariant I1: All extended ending nets in a block are positioned when they enter that block.

Phase 1.2b again enforces I1, by the arguments presented in Section 4.1.

Invariant I2: At most $L-2$ falling and $L-2$ rising fully positioned nets become unpositioned in a block.

Again, no nets become unpositioned during Phase 1.1 or Phase 1.2b. In Phase 1.2a, each of the $L-2$ runs of bad lane jogs moves only semi-positioned nets, except possibly the last net in the run.

Invariant I3: At most $4r$ falling and $4r$ rising extended nets enter any block.

As before, assume that x_f falling extended nets enter B, and e_f is the number of *new* falling ending nets in B. Then at least $\frac{(L-3)x_f}{L-2}$ falling extended nets are fully positioned, and by I2, at most $L-2$ of these become unpositioned in B. If $e_f \le \frac{(L-3)x_f}{L-2} - (L-2)$, we are done, so assume that $e_f > \frac{(L-3)x_f}{L-2} - (L-2)$. Then the number of extended nets leaving B is at most $x_f + e_f - \frac{(L-3)x_f}{L-2} + (L-2) = \frac{x_f}{L-2} + e_f + (L-2)$. Since $x_f \le 4r$, $e_f \le r$, $L \ge 3$, and r will be chosen $\ge L$, this quantity is at most $4r$.

The following lemmas for the multiterminal net case follow (the proofs parallel those given for the two-terminal net case).

Lemma 6: Phase 1.1 of the L-layer, multiterminal net algorithm uses at most $\frac{d}{L-2} + O(r/L)$ tracks for $r \ge L$.

Lemma 7: Phase 1.2 of the L-layer, multiterminal net algorithm requires at most $O((d/r)+1)$ tracks.

Lemma 8: Phase 2 of the L-layer, two-terminal net algorithm uses at most $O(r/L)$ tracks.

The total channel width is therefore $\frac{d}{L-2} + O(r/L) + O((d/r)+1)$ tracks. The block size, r chosen to minimize this quantity is again $r = O(\sqrt{dL}+1)$. Our algorithm applies when $L \geq 4$. If $L = 3$, the multiterminal net algorithm of [BB86] uses only d tracks. Thus, we have the following theorem.

Theorem 3: Any multiterminal net channel can be routed in the L-layer arbitrary overlap model using $\frac{d}{L-2} + O(\sqrt{d/L}+1)$ tracks, for $L \geq 3$.

5. Lower Bounds

5.1. Two-Layer Routing

For two-terminal net problems, the algorithms use $O(\sqrt{d/L}+1)$ tracks more than the simple worst case lower bound of $\frac{d}{L-1}$ tracks. It is tempting to conjecture that these extra tracks are not really necessary. For example, it might even seem more reasonable that only d tracks are required for two-terminal nets in the unit vertical overlap model. It is shown in what follows, however, that this is not the case. In particular, a worst case lower bound of $d + \Omega(\log d)$ tracks is proven for the two-layer case, therefore some function of d extra tracks is definitely necessary. The lower bound holds even if vertical overlaps of arbitrary length are allowed.

The lower bound argument proceeds as follows. Define a CRP consisting of four consecutive groups of d columns each, such that the d upper terminals of group i connect to the d lower terminals of group $i+1$, for $1 \le i \le 3$ (see Figure 15). The actual permutation of lower terminal connections within each group will be defined later. Notice that all of the nets are falling, and the density in columns $d+1$ through $3d$ is precisely d. Assume that $d+K$ tracks are available for routing. Of course, it is possible to solve some CRPs of this form when $K=0$. However, it will be shown in what follows that K must be at least $\Omega(\log d)$ for *most* CRPs with the specified form. The bound follows from a counting argument.

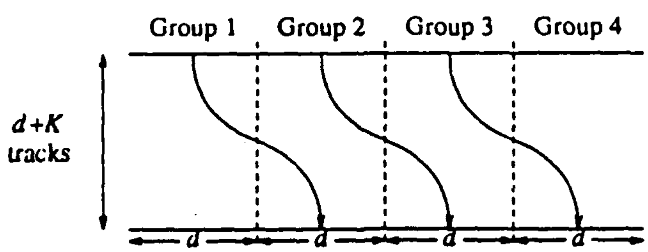


Figure 15. CRP for lower bound argument.

Consider a horizontal (vertical) cut between two tracks (columns). A net with a terminal on each side of the cut must use one or more unit vertical (horizontal) segments to cross the cut. The first segment of a net used to cross a cut which separates the net's terminals, traveling along the net's path from its top terminal to its bottom terminal, is called an *initial crossing* of the cut. A cut which does not split a net's terminals may contain only noninitial crossings of the net. The counting argument hinges on initial crossings. Unit segments of the grid which are unused or which contain noninitial crossings are called *nonuseful* segments.

Consider only the routing within the middle $2d$ columns of the CRP. This $2d$-column subproblem is referred to as the *restricted region*. First, some useful lemmas about routing within this region are proved.

Lemma 9: Exactly d horizontal segments per column cut contain horizontal initial crossings in the restricted region.

Proof: The density is d at all columns in the region. The d nets which must cross a column cut must each account for exactly one initial crossing in that cut. Further, nets which need not cross a cut do not contribute any initial crossings, and there are exactly d initial crossings. □

Lemma 10: There are at most K unit vertical overlaps in any column for which both wire segments are initial crossings of the corresponding horizontal cut in the restricted region.

Proof: This is because each unit vertical overlap causes a nonuseful horizontal segment on each side of the column, and no column may contain more than K of these. Consider a column segment in column c containing two initial crossings, as in Figure 16. The arrows indicate the direction of the net's path from its top terminal to its bottom terminal. Note that vertical useful segments are directed from top to bottom and horizontal useful segments are directed from left to right. Consider the lower left adjacent horizontal segment (see Figure 16a). It must be directed to the left, and therefore is not useful. Similarly, the upper right horizontal segment must also be left directed and nonuseful (see Figure 16b). □

Lemma 11: At most $4dK + K^2 + K$ vertical segments of the restricted region may fail to contain an initial crossing.

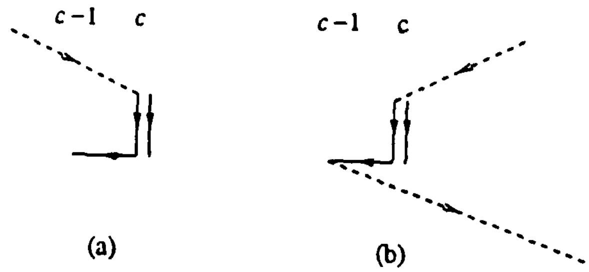


Figure 16. Routing characteristics near unit vertical overlaps.

Proof: At best, the leftmost d nets can enter the restricted region on the bottommost d tracks. In this way, the net on track 1 requires only one vertical initial crossing in the restricted region, the net in track 2 requires two vertical initial crossings, and so forth. In all, $1+2+3+ \ldots +d = \frac{(d+1)d}{2}$ vertical initial crossings result. Similarly, the rightmost d nets use at least another $\frac{(d+1)d}{2}$ vertical initial crossings. Only K horizontal segments on each side of the restricted region are not used by the leftmost and rightmost d nets. Thus, at most K of the d middle nets can exit and reenter the restricted region ($K/2$ on each side). The remaining $d-K$ middle nets must be routed entirely within the region, and each contributes $d+K+1$ vertical initial crossings to this region. Thus, the restricted region contains at least $(d-K)(d+K+1) + d(d+1)$ vertical initial crossings in all. There are a total of $2d(d+K+1)$ vertical segments available in the restricted region, and by Lemma 10, at most $2dK$ of them can contain two initial crossings (using vertical overlap). Thus, at most $[2d(d+K+1) + 2dK] - [(d-K)(d+K+1) + d(d+1)] = 4dK + K^2 + K$ vertical segments do not contain an initial crossing. □

The initial strategy is to bound the number of different choices allowed for routing a single column. The order in which the leftmost d nets enter the leftmost column of the restricted region will be decided by the choice of routings made within the region, so that the correct connections are made to the lower left terminals. Assume that the restricted region contains E nonuseful vertical segments, spread among the $2d$ columns. By Lemma 11, $E \leq 4dK+K^2+K$. Define each column i to have k_i nonuseful vertical segments, such that $\sum_{i=1}^{2d} k_i = E$. Since E nonuseful vertical segments are to be divided among $2d$ columns, the number of possibilities for the $\{k_i\}$ is bounded by $\binom{E+2d+1}{E} \leq 2^{E+2d}$.

We define the routing of a single column c to consist of assigning wires to the $d+K+1$ vertical segments in c and to the $d+K$ horizontal segments of the column cut $(c,c+1)$. The routing of the horizontal segments of column cut $(c-1,c)$ have already been fixed when the routing in column c is considered. There are four different ways to route a vertical segment: empty, layer 1 wire, layer 2 wire, or both (a unit vertical overlap). There are only three possibilities for horizontal segments, since horizontal overlap is not allowed. Thus, there are $4^{d+K+1}\,3^{d+K}$ different ways to specify a column routing, although many of the possibilities do not correspond to a legal routing. We show that in the restricted region the number of different ways to route a column is far more limited.

Each column cut i can be divided into $K+1$ groups of consecutive horizontal initial crossings, separated by up to K nonuseful horizontal segments. Routing within a group of horizontal initial crossings is restricted in the following sense. Look at the highest pair of two consecutive vertical initial crossings in the group, as shown in Figure 17a. We draw the initial crossing segments with arrows indicating the direction of a path from the upper terminal to the lower terminal. We show that all of the routing within the group below this pair of vertical segments is uniquely determined.

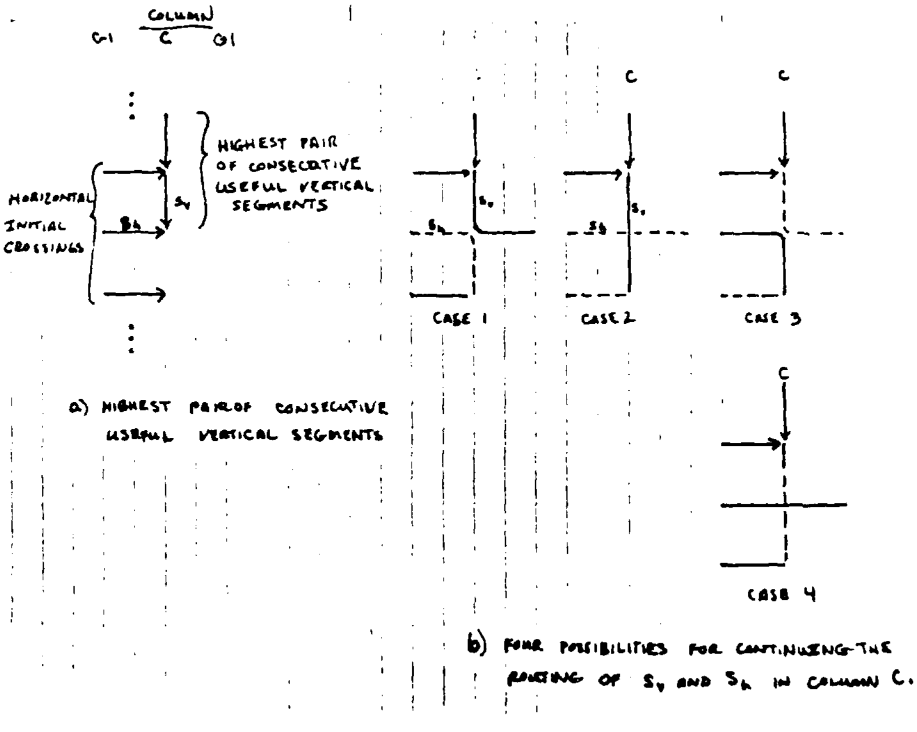


Figure 17. Routing below the highest two consecutive useful vertical segments within a group.

Consider the lower of the two vertical initial crossing segments in c (call it s_v) and the horizontal initial crossing (s_h) from cut $(c-1,c)$ which is incident upon its lower endpoint. Since both nets are directed toward their incident point, they do not belong to the same net. Note that s_h cannot turn upward, since there are two initial crossings directed into the point above already. The vertical wire, s_v, cannot turn left since horizontal overlap is not allowed. For the same reason, at most one of the two nets can continue to the right. Finally, the point below contains an inward directed horizontal initial crossing from column $c-1$, so both s_v and s_h cannot be routed downward. Thus, one of the nets is routed down, and the other to the right. Further, the net which is routed downward must not be in the same layer as the horizontal segment entering from the left. There are only four ways for the routing at such a point to occur, shown in Figure 17b. Note that the layer of s_h and the horizontal crossing immediately below s_h uniquely determines the routing at this point.

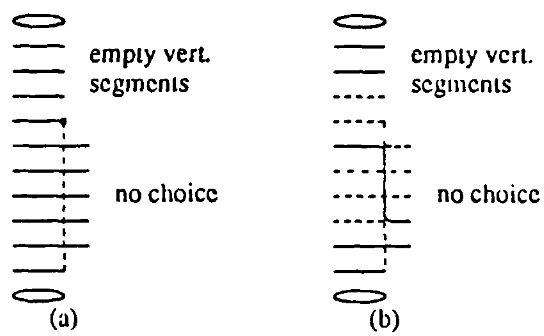


Figure 18. Restricted routing within a group.

Now, consider the next lower point. We have shown that in all four cases, a downward directed wire enters the point from above, so we can apply the same argument to show that the routing at this point is also uniquely determined. This argument is applied in order to each point of the group below s_v and s_h, proving that this portion is fixed. This claim is illustrated by the examples in Figure 18. Now look at the area from the top of the group to the highest pair of consecutive vertical initial crossings. At most half of the vertical segments in this area contain initial crossing segments, since no two are consecutive. But the entire column contains only k_i nonuseful vertical segments, so the size of this area is bounded. The area consists of $2k_i$ vertical segments and $2k_i$ horizontal segments which are not uniquely determined. Also, there are K vertical segments and K horizontal segments corresponding to the continuation of the routing at the K nonuseful horizontal segments of cut $(c-1,c)$. Further, there are $K+1$ horizontal and vertical segments corresponding to the top half of the highest pair of horizontal initial crossings in the group, i.e., the track above s_k to be routed. There are a total of at most 4^{2k_i+2K+1} ways to route all the vertical segments and 3^{2k_i+2K+1} ways to route all the horizontal segments.

Finally, the *location* of these segments must be determined. The location depends on how the k_i vertical nonuseful segments are distributed among the $K+1$ groups. There are at most 2^{K+1+k_i-1} ways to divide the k_i nonuseful segments between the $K+1$ sets of horizontal initial crossings. So column i can be routed in at most $4^{2k_i+2K+1}\,3^{2k_i+2K+1}\,2^{K+1+k_i-1} < 2^{9k_i+9K+4}$ different ways.

Now consider the entire region of $2d$ columns. The maximum number of different routings for the entire region is $\prod_{i=1}^{2d} 2^{9k_i+9K+4} = 2^{9E+(9K+4)2d}$, since $\sum_{i=1}^{2d} k_i \le E$. Furthermore, $E \le 4Kd + K^2 + K \le 6Kd$, assuming without loss of generality, that $K \le d$. But at least $d!$ different problems must be routable in the region, corresponding to $d!$ different permutations of the lower terminals of the nets in set 2. Thus, $2^{72Kd+8d} \ge d! > \left(\frac{d}{e}\right)^d$ (by Stirling's approximation), and so $K = \Omega(\log d)$.

Theorem 4: The optimal solution to some CRPs of density d allowing vertical overlap requires $d + \Omega(\log d)$ tracks.

Note that this lower bound proof counts only the *number* of different routings possible in a given area. It may grossly overestimate the number of different problems that can be routed, since there are many different ways in which to route a single problem. Also, the problem instance considered is not at all complex. It contains only $4d$ nets, all of them falling. Thus, it seems feasible that the lower bound could be further improved using a different strategy. We suspect that in the worst case, $d + \Omega(\sqrt{d})$ tracks may actually be required.

5.2. Extension to Three or More Layers

The lower bound argument extends to $L \ge 3$ as well. Consider an L-layer model in which at most $L-1$ layers may overlap horizontally, while all L layers may overlap in the vertical direction. Given the same initial problem, attempt to route it in $\frac{d+K}{L-1}$ tracks. By the preceding arguments, in each column there are at most $K+1$ sets of consecutive tracks which each contain $L-1$ horizontal initial crossings. As before, there is no freedom in routing these sets below the first pair of downturning nets.

The counting parallels the previous two layer case. First, note that d horizontal lane segments contain a horizontal initial crossing in the restricted region (a horizontal (vertical) segment contains L horizontal (vertical) lane segments). Next, note that at any point, the number of vertical initial crossing lane segments entering from the top plus the number of horizontal initial crossing lane segments entering from the left is at most L. Thus every *extra* vertical initial crossing (i.e. every one except the first in a vertical segment) causes a nonuseful horizontal lane segment. There are at most K nonuseful horizontal lane segments at any column cut, therefore there at most K extra vertical initial crossings per column.

Next, we count the number of nonuseful vertical segments (vertical segments which contian no initial crossings at all). The restricted region contains $2d(\frac{d+K}{L-1}+1)$ total vertical segments, and there are at most $2dK$ extra vertical initial crossing lane segments. The restricted region contains at least $(d-K)(\frac{d+K}{L-1}+1) + d(\frac{d+K}{L-1}+1)$ vertical initial crossings, so the number of nonuseful vertical initial crossings is $\leq [\, 2d(\frac{d+K}{L-1}+1) + 2dK \,] - [\, (d-K)(\frac{d+K}{L-1}+1) + d(\frac{d}{L-1}+1) \,] = \frac{2dK+K^2}{L-1} + K + 2dK \leq 4dK+K^2+K$. This quantity is bounded by $O(dK)$ for $K \leq d$. The number of ways to route the L lanes of a segment is bounded by $2^{O(KdL)}$, so we have $2^{O(KdL)} \leq \left(\frac{d}{e}\right)^d$, and therefore $K \geq \Omega\left(\frac{\log d}{L}\right)$, giving the following theorem.

Theorem 5: The optimal L-layer solution to some CRPs of density d, allowing $L-1$ layers to overlap horizontally and all L layers to overlap vertically, requires $\frac{d}{L-1} + \Omega\left(\frac{\log d}{L^2}\right)$ tracks.

6. Remarks and Open Questions

The work presented here has served to tie together many of the theoretical channel routing results and has resolved a number of open issues. There are two central questions left open by this work. First, is the additive $O(\sqrt{d})$ term necessary? We conjecture that it is, in the worst case, even for the least restrictive two-layer model. In support of the conjecture, we have shown that some CRPs require $d + \Omega(\log d)$ tracks when vertical overlap of arbitrary length is allowed.

Second, do multiterminal net problems really require larger channel widths than two-terminal net problems? For almost all models studied, the best known algorithms have in the worst case used roughly a factor of two times as many tracks for multiterminal as for 2-terminal nets (nonadjacent overlap models being the notable exceptions). As mentioned in Section 2, Gao and Kaufmann succeeded in reducing the multiterminal net upper bound for unit vertical overlap, L=2, to $3d/2 + O(\sqrt{d \log d})$. We have shown that this barrier breaks down for $L > 3$.

The algorithms presented in this paper are all based on the same ideas and therefore give a unified framework in which to obtain bounds for various routing models: Manhattan, knock-knee, two-layer unit vertical overlap, routing with 45-degree diagonal wires, and multilayer routing with arbitrary overlap. Thus, we can hope that an improvement of a bound for any of these models might well generalize to the others. Finally, we expect to be able to improve the constants in our algorithmic bounds, since we have presented only the important concepts here, rather than attempt to use the absolute fewest number of tracks.

Acknowledgement

We are indebted to Ron Pinter for valuable ideas and discussions.

References

[BBL84] Baker, B., S. N. Bhatt, and T. Leighton, "An Approximation Algorithm for Manhattan Routing", *Advances in Computing Research 2 (VLSI Theory)*, ed. F. P. Preparata, JAI Press, Inc., Greenwich, CT, pp. 205-229 (1984).

[Be86] Berger, B., "New Upper Bounds for Two-Layer Channel Routing", M.S. thesis, Massachusetts Institute of Technology (Jan. 1986).

[Br87] Brady, M., "New Algorithms and Bounds for Multilayer Channel Routing", Ph.D. thesis, University of Illinois at Urbana-Champaign (May 1987).

[BB82] Bolognesi, T., D. J. Brown, "A Channel Routing Algorithm with Bounded Wire Length", unpublished manuscript, Coordinated Science Lab., University of Illinois at Urbana-Champaign, (1982).

[BB86] Brady, M., D. J. Brown, "Optimal Multilayer Channel Routing With Overlap", *Proc. 4th MIT Conf. on Advanced Research in VLSI*, pp. 281-296 (Apr. 1986). (To appear in *Algorithmica.*)

[BR81] Brown, D. J., R. L. Rivest, "New Lower Bounds for Channel Width", *Proc. 1981 CMU Conf. on VLSI Systems and Computations*, pp. 178-185 (Oct. 1981).

[De76] Deutsch, D. N., "A Dogleg Channel Router", *Proc. 13th Design Automation Conf.*, pp. 425-433 (June 1976).

[Fr82] Frank, A., "Disjoint Paths in a Rectangular Grid", *Combinatorica* 2 (4), pp. 361-371, (1982).

[GH86] Gao, S., S. Hambrusch, "Two-Layer Channel Routing with Vertical Unit-Length Overlap", *Algorithmica*, 1 (2), pp. 223-232, (1986).

[GK87] Gao, S., M. Kaufmann, "Channel Routing of Multiterminal Nets", *Proc. 28th IEEE Symp. on Foundations of Computer Science*, pp. 316-325 (Oct. 1987). (Submitted to *JACM.*)

[Ha83] Hambrusch, S., "Using Overlap and Minimizing Contact Points in Channel Routing", *Proc. 21st Annual Allerton Conf. on Communication, Control, and Computing*, pp. 256-257 (Oct. 1983).

[Ha85] Hambrusch, S., "Channel Routing Algorithms for Overlap Models", *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, CAD-4 (1), pp. 23 30 (Jan. 1985).

[HS71] Hashimoto, A., J. Stevens, "Wire Routing by Optimizing Channel Assignment Within Large Apertures", *Proc. 8th Design Automation Workshop*, pp. 155-169, (1971).

[Le82] Leighton, F. T., "New Lower Bounds for Channel Routing", MIT VLSI Memo 82-71, (Jan. 1982). (Submitted to *SIAM J. Discrete Math.*)

[LLP87] Lodi, E., F. Luccio, L. Pagli, "A Preliminary Study of a Diagonal Channel Routing Model", to appear in *Algorithmica.*

[PL84] Preparata, F. P., W. Lipski, Jr., "Optimal Three-Layer Channel Routing", *IEEE Trans. on Computers*, C-33, pp. 427-437 (1984).

[RBM81] Rivest, R. L., A. Baratz, and G. Miller, "Provably Good Channel Routing Algorithms", *Proc. 1981 CMU Conf. on VLSI Systems and Computations*, pp. 153-159 (Oct. 1981).

[Ri82] Rivest, R. L., "The PI (Placement and Interconnect) System", *Proc. 19th IEEE Design Automation Conf.*, pp. 475-481 (June 1982).

[RF82] Rivest, R. L., and C. M. Fiduccia, "A Greedy Channel Router", *Proc. 19th Design Automation Conf.*, pp. 418-424 (June 1982).

[SP85] Sarrafzadeh, M., F. P. Preparata, "Compact Channel Routing of Multiterminal Nets", *Annals of Discrete Math.* 25, pp. 255-280 (1985).

[Sa87] Sarrafzadeh, M., "Hierarchical Approaches to VLSI Circuit Layout", Ph.D. dissertation, University of Illinois at Urbana-Champaign (Jan. 1987).

[SY82] Szymanski, T. G., and M. Yannakakis, private communication.

[Sz85] Szymanski, T. G., "Dogleg Channel Routing is NP-Complete", *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, CAD-4 (1), pp. 31-41 (Jan. 1985).

[YK82] Yoshimura, T., and E. S. Kuh, "Efficient Algorithms for Channel Routing", *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems* CAD-1 (1), pp. 25-35 (Jan. 1982).